ORIC

GUIDE PRATIQUE DE L'ORIC-ATMOS

Michel Bussac
et Robert Lagoutte

du Basic à l'Assembleur



cedic/nathan

GUIDE PRATIQUE DE L'ORIC - ATMOS

du Basic à l'Assembleur

Michel Bussac et Robert Lagoutte

Dessins : Dominique Carrara

cedic/nathan

Dans la même collection

Initiation au BASIC TO7 — *Christine et François-Marie Blondel.*
Le BASIC D.O.S. du TO7/TO7.70 et du MO5 — *Christine et François-Marie Blondel.*
Un ordinateur à la maison — *Jean Delcourt.*
Un ordinateur en fête — *Serge Pouts Lajus.*
Un ordinateur et des jeux — *Jean-Pascal Duclos.*
Guide pratique de l'ordinateur personnel IBM — *Dalloz - Emery - Portefaix - Boisgontier-Salzman.*
LOGO, des ailes pour l'esprit — *Horacio C. Reggini (traduction de E. et S. Letelier et A. Schnake).*
Premiers pas avec le ZX-SPECTRUM — *Ian Stewart et Robin Jones (adaptation de Claire Touchard).*
Le Langage machine du ZX-SPECTRUM — *Ian Stewart et Robin Jones (adaptation de David Hartley et Nadia Arcas).*
Plus loin avec le ZX-SPECTRUM — *Ian Stewart et Robin Jones (adaptation de Claire Touchard).*
Jeux vidéo, Jeux de demain — *Georges-Marie Becherraz et Alain Graber.*
Guide pratique de l'Oric-Atmos — *Michel Bussac et Robert Lagoutte.*
Des programmes pour votre Oric — *Michel Piot*
Premiers pas avec le Commodore 64 — *Ian Stewart et Robin Jones (adaptation de Claire touchard).*

Ce volume porte la référence
ISBN 2-7124-0528-5 (1re publication)
Édition 1984, revue et corrigée ISBN 2-7124-0585-4

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS

Sommaire

Deuxième partie: ATMOS en fiches

Annexes :

AVERTISSEMENT AU LECTEUR

Ce GUIDE PRATIQUE de l'ATMOS est une édition renouvelée de celui de l'ORIC 1. Il a été corrigé, adapté aux caractéristiques techniques de l'Atmos et enrichi d'un nouveau chapitre, de nouveaux programmes ainsi que d'informations annexes importantes.

Avant-propos

Oric ATMOS actuellement le plus puissant dans sa gamme, possède un BASIC souple, riche en possibilités graphiques et sonores. Que vous soyez débutant ou programmeur confirmé, sa découverte vous procurera les joies d'un voyage sidéral.

Installez-vous aux commandes, votre GUIDE PRATIQUE à portée de la main. Vous constaterez que cet ouvrage a été conçu pour vous permettre d'accéder efficacement aux informations.

Si vous êtes nouveau-venu dans le monde des ordinateurs ou si le langage BASIC ne vous est pas familier, vous trouverez dans la première partie, LE PILOTAGE DE L'ATMOS, une initiation progressive, accompagnée d'exemples. De "l'embarquement" à la "croisière hyperbasic", vous explorerez l'espace mémoire jusqu'au monde de l'Assembleur. Là, vous effectuerez vos "premiers pas en langage machine". En cours de route, vous pourrez faire le point grâce au "Check-List" de chaque chapitre et vous répérer aisément à l'aide du LEXIQUE GÉNÉRAL en fin d'ouvrage.

La deuxième partie, ATMOS EN FICHES, présente des fiches de référence. Classées par thème, faciles à consulter, elles décrivent l'ensemble des possibilités d'ATMOS. Elles sont suivies d'ANNEXES contenant des précisions techniques et pratiques.

Note : le livre peut être vendu avec ou sans cassette d'accompagnement. Cette cassette contient les vingt principaux programmes de jeux, vie pratique, gags, musique et dessins dont la liste figure à la fin de cet ouvrage. Son usage n'est pas obligatoire mais vous évitera les recopiages fastidieux de listes d'instructions et facilitera votre décollage.

Première partie

Le pilotage de l'ATMOS

Embarquement sur ATMOS

1. Prise en mains

On peut comparer votre ATMOS à une machine spatiale. Les différentes parties qui la font "tourner" sont les suivantes :

— Le cœur de la machine s'appelle l'*unité centrale.* On ne voit pas ce véritable "moteur" mais c'est lui qui dirige tout, sous votre contrôle, bien sûr. L'unité centrale se trouve à l'intérieur de la coque plastique de l'ATMOS. Elle comprend un microprocesseur 6502A, une mémoire de 64 K et des circuits électroniques annexes.

— *Le clavier :* c'est le poste de commande à partir duquel vous donnerez vos ordres à l'unité centrale.

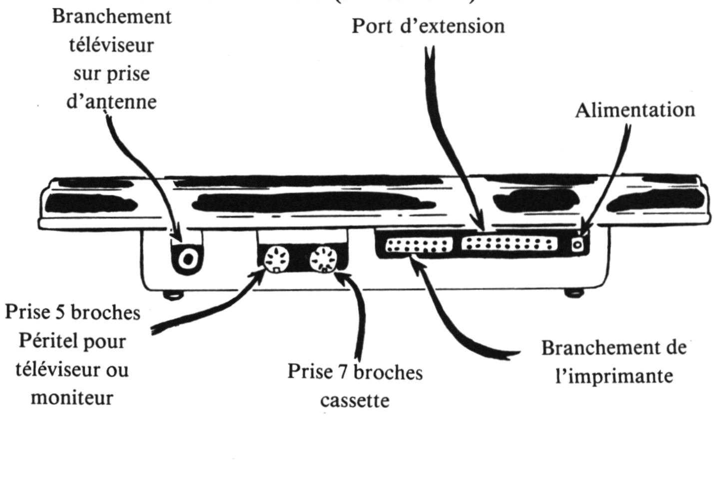
— *L'écran.* Pour voir l'univers et ses planètes en couleur, utilisez un téléviseur avec prise *PERITEL.* C'est avec lui que vous échangerez des messages avec l'ATMOS.

— Le *magnéto-cassette* est le réservoir pour stocker ou puiser les informations que consomme l'ATMOS.

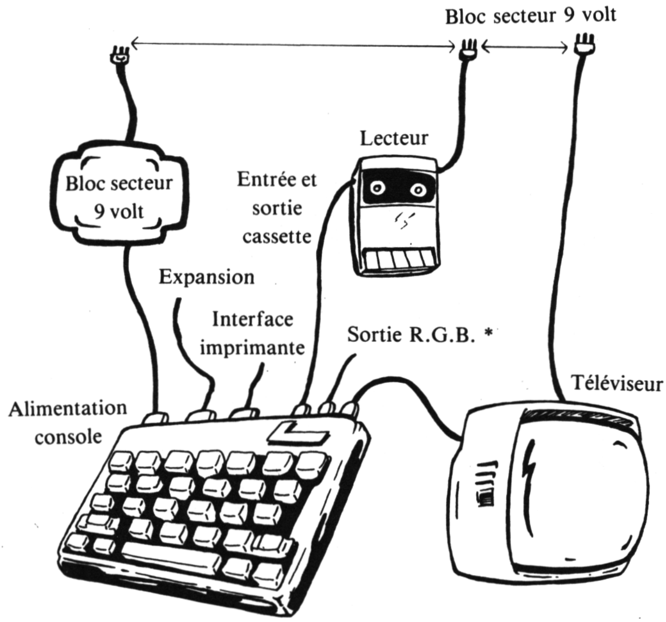
Tous ces organes doivent être reliés entre eux par des cordons. Sans oublier, bien entendu, les alimentations électriques nécessaires au "décollage".

2. Mise en marche

Le cable noir, avec prise PERITEL (grosse prise noire à 20 broches), relie le téléviseur par sa propre prise PERITEL à la prise RVB (prise DIN à 5 broches) située à l'arrière de votre ATMOS (cf. Schéma).



Sur la prise PERITEL du cable, vous devez en plus brancher une alimentation 12 VOLTS, fournie avec l'appareil : il s'agit du plus petit des deux blocs-secteur en plastique noir. N'oubliez pas de régler son commutateur sur 12 volts et **surtout** de vérifier la correspondance des + et des – du boitier et de la petite fiche.
Le gros bloc secteur sera relié à la petite prise ronde, située à l'arrière et à l'extrême gauche du clavier.



* Sortie R.G.B. pour branchement Péritel sur téléviseur couleur

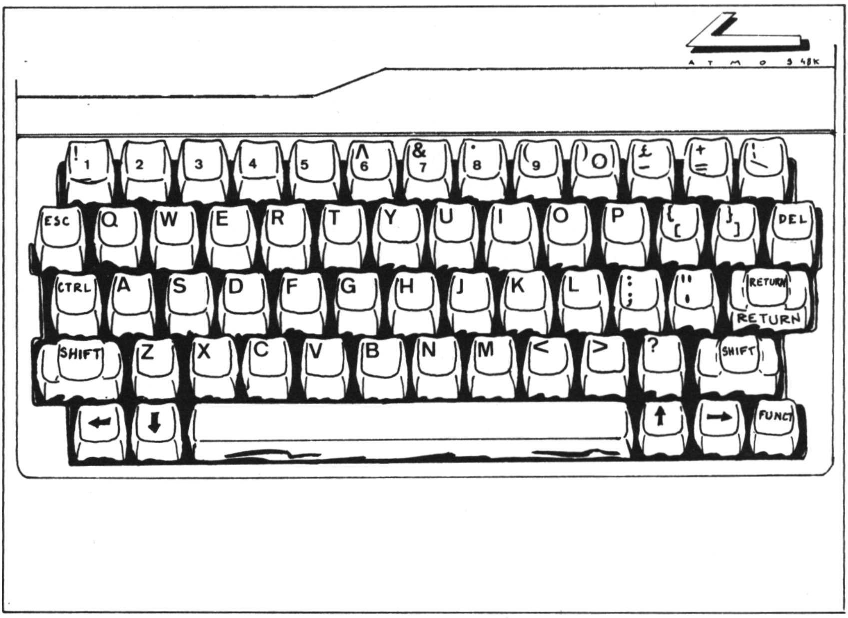
Schéma général de la configuration

La prise DIN du magnétophone est reliée à la prise DIN à 7 broches de l'ATMOS. Côté magnétophone, si vous avez un cable standard, sachez que le cordon rouge est relié à l'entrée MICRO. Un conseil, réglez les potentiomètres à 80 % du maximum.
Une fois effectuées ces connections, branchez les prises de courant (220 volts).

L'écran de votre téléviseur se couvre de rayures horizontales puis apparaît l'inscription :
ATMOS EXTENDED BASIC V1.1
(C) 1983 TANGERINE
XXXX... BYTES FREE (1)
ready

Si ce message n'apparaît pas et que les rayures persistent, débranchez la prise secteur sur l'arrière de l'appareil et rebranchez-la après quelques secondes.
Vous voilà prêt à prendre les commandes.

3. Clavier, écran et commandes élémentaires



Vous avez sous les yeux les 57 touches de l'ATMOS. Essayons d'y voir plus clair en distinguant quatre ensembles de touches :
— *Au centre :* 47 touches de lettres, chiffres, ponctuations, espacement qu'on retrouve sur toutes les machines à écrire.
— *A gauche :* [ESC], [CTRL], [SHIFT], trois touches dont le rôle est important. Elles contrôlent et peuvent modifier le fonctionnement des précédentes. Remarquons que la touche [SHIFT] a une sœur jumelle à droite du clavier.

(1) Le terme BYTES FREE indique la quantité de mémoire disponible. Dans la version 16 K, ATMOS offre 4863 Bytes et dans la version 48 K, 37631 bytes (octets en français). Se reporter au chapitre consacré à la gestion de la mémoire.

— *De part et d'autre de la barre d'espacement,* on trouve 4 touches marquées par des flèches : elles servent au déplacement du carré clignotant qu'on appelle le "curseur".
Vous pouvez déjà vous entraîner à déplacer le curseur sur l'écran.
— *A droite,* deux touches particulères [RETURN] et [DEL]. Détaillons-en le rôle en les manipulant :

Premiers affichages sur l'écran

Tapez sur le clavier, par exemple, "BONJOUR". Vous observez que sur l'écran s'inscrivent les caractères mais qu'il ne se passe rien de plus.
En fait, ATMOS se contente d'emmagasiner les caractères successifs. Si vous voulez que la machine prenne votre message, il faut le lui dire. Sinon, elle ne saura jamais que votre message est terminé.
C'est le rôle de la touche [RETURN] qui signifie "prends ce que je viens de taper et fais-en ce que tu peux". (Sur certaines machines, elle s'appelle ENTER).
Tapez :
[RETURN]
? SYNTAX ERROR
ready (répond la machine).
La première ligne signifie : "je ne comprends pas ce que vous me dites : revoyez le dictionnaire ou la grammaire". La deuxième : "je suis prêt". Conclusion : le mot "BONJOUR" n'est pas connu de la machine.
Essayez :
ATMOS
[RETURN]
? SYNTAX ERROR
ready
Cette machine ne semble pas connaître son propre nom. A moins que nous ne sachions pas lui parler. Jetons un coup d'œil sur la liste de "son" vocabulaire et essayons :
PRINT "ATMOS"
[RETURN]
ATMOS
ready
(pour obtenir les guillemets appuyez sur [SHIFT] en même temps que sur la touche guillemets")
Nous reviendrons plus tard sur le mot PRINT. Mais il est clair qu'ATMOS, comme tout ordinateur, ne connaît qu'un certain nombre de mots et qu'il n'admet aucune faute de votre part.
Si vous faites une erreur en tapant un caractère, vous pouvez l'effacer en tapant sur [DEL]
DEL est, en somme, le contraire de RETURN.
Mais attention, DEL efface toujours le caractère situé à gauche du curseur. De plus, l'effacement n'est effectif (enregistré par la machine) que pour le caractère que l'on vient de taper.

Par exemple, supposons que nous ayons tapé, par erreur :
APRINT "BONJOUR"
Ramenons le curseur après le A et tapons [DEL]. Puis amenons le curseur à droite : la phrase est correcte. Mais appuyons sur [RETURN] :
SYNTAX ERROR

Nous constatons que DEL n'a corrigé que sur l'écran : pour être effective la correction aurait dû être faite immédiatement après l'erreur.
Après quelques exercices du même genre, vous saurez vite écrire des mots, effectuer des corrections.
Si votre écran est déjà barbouillé par vos premiers essais, passez un coup d'éponge grâce à CLS. Cette instruction vide l'écran et replace le curseur en haut et à gauche.

Essayez : tapez CLS et [RETURN].

Autre découverte : la répétition d'un même caractère si l'on maintient la pression sur la même touche. Essayez : appuyez sur la touche de votre choix. Si vous insistez la répétition se produit. Très pratique pour déplacer le curseur rapidement sur l'écran.

La répétition nous permet de constater qu'ATMOS n'accepte qu'à peine plus de deux lignes de caractères à la suite. Très exactement 38 caractères par lignes plus 3 caractères dans la troisième ce qui fait en tout 79. Il a la politesse de nous prévenir par une sonnette, dès le 77e, que nous dépassons les limites autorisées.

Vous avez certainement remarqué que certaines touches possèdent deux significations. Vous savez, sans doute aussi, qu'on peut écrire en minuscules. Et la couleur, vous demandez-vous. Et bien, tout cela va être mis à votre portée grâce aux *touches de contrôle* que nous allons maintenant étudier.

Touches de contrôle

[SHIFT] est présente à gauche et à droite. Son rôle est de donner la valeur du haut pour les touches comportant deux indications. Essayons ensemble : appuyons sur un SHIFT et — sans relâcher — sur l'autre touche [?/], par exemple. Cela donne / sur l'écran.
[SHIFT] [?/] ?
[6^] 6
[SHIFT] [6^] ↑

[CTRL] actionnée simultanément avec une autre touche, donne un caractère de contrôle. Non visible sur l'écran, ce caractère a un effet sur le fonctionnement de l'ordinateur.
Voici les principales commandes :
[CTRL] [A] sert à la copie d'un caractère à la position du curseur (Voir la partie consacrée à l'éditeur p. 55)

[CTRL] [T] Passage du mode majuscule au mode minuscule et inversement. En essayant, vous verrez apparaître ou disparaître le message CAPS (pour capitales) en haut et à droite de l'écran. CAPS signale l'utilisation des capitales pour une raison importante : rappeler que les instructions en BASIC doivent être écrites en CAPITALES sinon elles ne sont pas acceptées.
[CTRL] [Q] fait disparaître le curseur. Activez de nouveau cette commande pour le revoir.
[CTRL] [X] fait sauter une ligne.
[CTRL] [F] Supression du Bip du clavier.
[CTRL] [L] Effacement de l'écran et retour du curseur en haut et à gauche.
[CTRL] [C] Arrêt d'un programme BASIC en cours d'exécution.
[CTRL] [G] Sonnette
Il en existe d'autres que vous pourrez découvrir en annexe numéro cinq.

Nous avons travaillé, jusqu'à présent en noir et blanc. Un peu triste n'est-ce pas ?
Vous pouvez, dès à présent, modifier les couleurs de l'écran et de l'écriture grâce aux commandes suivantes :
PAPER 4 (nous n'écrirons plus systématiquement [RETURN], cela doit devenir automatique).
Le fond devient bleu foncé.
INK 1 : les caractères rougissent...
Voici la palette, à vous de jouer avec ce code :

Ø Noir
1 Rouge
2 Vert
3 Jaune
4 Bleu
5 Magenta
6 Cyan
7 Blanc
Rappelez-vous : PAPER suivi du code fixe la couleur du fond, INK suivi du code, celle des caractères.
Essayez les combinaisons et choisissez celle que vous préférez. Vous pouvez en changer tous les jours. Et... n'oubliez pas RETURN !

[ESC] C'est un peu plus compliqué. Nous y reviendrons en étudiant le graphique. Essayez tout de même :
ESC A puis ESC T sur une ligne de texte.
Vous constaterez que les caractères et le fond ont changé de couleur.
[ESC] permet de modifier la présentation d'une ligne de l'écran.
Et puis, avant le décollage en Basic, un peu d'ambiance sonore voulez-vous ? ATMOS possède quelques commandes assez sophistiquées que nous étudierons en détail. Voici quelques échantillons de ce qui vous attend dans l'espace sonore d'ATMOS.
Écrivez ZAP sur le clavier, après validation vous obtiendrez une décharge sifflante digne de la guerre des étoiles.

Recommencez avec EXPLODE, PING et SHOOT : vous voilà dans l'ambiance du départ.



4. Quatre programmes...

Nous vous recommandons de les taper, à titre d'entraînement à l'utilisation du clavier.
A cette occasion, vous vous familiariserez avec quelques commandes de base ainsi qu'avec l'emploi du magnéto-cassette et de l'imprimante.
Nos conseils : reproduisez fidèlement ces programmes ! ATMOS est un amoureux de l'exactitude. En cas d'erreur, retapez la ligne entière : elle se replacera automatiquement dans le programme.
Pour obtenir les guillemets n'oubliez pas SHIFT qui vous permet d'obtenir les caractères placé en haut des touches à double action.
Du courage, ce n'est pas long ! Et n'oubliez pas les [RETURN], en fin de ligne.

Starter numéro 1

```
1 CLS
10 REM STARTER 1
20 CLS
30 PAPER 2
40 INK 1
45 PRINT"QUELLE EST LA ";
50 INPUT "COULEUR DES CARACTERES";R$
```

```
60 IF R$<>"ROUGE" THEN 45
70 PRINT "BRAVO,VOUS N'ETES PAS DALTONIEN"
80 WAIT 300
90 PRINT
100 PRINT"DONNEZ DEUX NOMBRES SEPARES";
103 PRINT" PAR UNE VIRGULE"
107 INPUT X,Y
120 PRINT"LEUR SOMME EST ";X+Y
130 PRINT"LEUR PRODUIT EST ";X*Y
140 PRINT"LEUR QUOTIENT EST ";X/Y
```

Maintenant, à ATMOS de jouer : tapez RUN pour lancer le programme. Si ATMOS n'est pas satisfait de votre premier travail, il vous le fera savoir par le message SYNTAX ERROR : vérifiez, ligne d'instructions par ligne, la correction de votre frappe. Pour cela, tapez LIST et vous obtiendrez une nouvelle version listée de votre programme.

Le déroulement de l'exécution du programme peut être interrompu par [CTRL] [C]. Essayez après RUN : vous verrez apparaître le message BREAK IN LINE...

Sauvegarde du programme starter 1 sur cassette

STARTER 1 "tourne" sans bavure. Vous souhaitez pouvoir le réutiliser dans quelques jours. Votre MAGNÉTO-CASSETTE va vous servir de mémoire de conservation longue durée.

Assurez-vous qu'il est bien connecté. Appuyez sur la touche d'enregistrement, tapez CSAVE "STARTER1", S puis [RETURN].

Le nom que vous placez entre guillemets est désormais celui du programme actuellement dans la mémoire vive d'ATMOS. C'est sous ce nom qu'il le reconnaîtra désormais sur la bande magnétique de la cassette, sa mémoire "externe". Ce nom peut compter jusqu'à 17 caractères.

En haut de l'écran, apparaît le message "Saving STARTER 1" ce qui confirme que le programme est en cours d'enregistrement.

Lorsque c'est fini, le message "Ready" apparaît sur l'écran.

Starter numéro 2

Le premier programme étant recopié sur la cassette, vous voulez passer au second.

Pour ce faire, il vous faut d'abord effacer le premier de la mémoire vive d'ATMOS pour éviter un mélange d'instructions auquel il ne comprendrait rien !

Deux solutions : soit débrancher l'appareil ce qui vide la mémoire, soit utiliser la commande NEW qui produit le même effet. Testons la deuxième solution : tapez NEW puis [RETURN]

Note : Vous pourrez aussi taper : CSAVE "STARTER 1" et l'enregistrement aura lieu plus rapidement (mais votre magnéto-cassette peut alors provoquer plus d'erreurs de lecture).

Si vous tentez un RUN ou un LIST vous obtiendrez simplement un "Ready", preuve qu'aucun programme n'est en mémoire.
A présent vous pouvez taper le starter numéro deux :
10 REM...
Une simple REMarque : l'instruction REM que vous voyez au début des quatre starters n'a qu'un but, permettre d'insérer des titres, des commentaires ou des remarques pour rendre les listings plus lisibles. Cette instruction est sans effet sur le programme. Reprenons, si vous le voulez bien.

```
145 REM STARTER 2
150 PAPER 4
160 A$="BONJOUR ,MON NOM EST ORIC ET VOUS"
170 PRINTA$
180 INPUT NOM$
190 PRINT"BIENVENUE A BORD "NOM$
200 WAIT 200
210 PRINT
220 PRINT"VOTRE NOM CONTIENT ";
221 PRINTLEN(NOM$)" CARACTERES"
225 PRINT
230 PRINT"IL COMMENCE PAR UN: "LEFT$(NOM$,1)
235 PRINT
240 PRINT"...ET SE TERMINE PAR UN "RIGHT$(NOM$,1)
```

Chargement d'un programme
Vous souhaitez recharger STARTER1 ou STARTER2 dans la mémoire vive d'ATMOS, quelques jours plus tard.
Le magnéto-cassette correctement branché, appuyez sur la touche lecture de votre appareil, tapez CLOAD "STARTER1", S et [RETURN]. (Ou CLOAD "STARTER 1" si la sauvegarde a été faite en enregistrement rapide).
Un message "SEARCHING" (en recherche) s'affiche en haut de l'écran. Dès que le programme est trouvé, un autre message le remplace : "LOADING".
Dans le cas où vous auriez oublié le nom de votre programme, tapez simplement : CLOAD"""",S (1). Le programme chargé sera le premier rencontré. Si ce n'est pas le bon, recommencez l'opération de chargement.
Ces commandes peuvent varier selon la vitesse de sauvegarde ou de chargement choisie. Pour davantage de précision, reportez-vous à la fiche de référence : commandes du magnéto-cassette.

Starter numéro 3

Ce programme "tourne" mais vous pouvez introduire un arrêt automatique dans son déroulement, en introduisant une ligne STOP. Par exemple, ajoutez :
365 STOP

---

(1) Ne laisser aucun blanc entre les guillemets.

A l'exécution, le programme s'interrompt en ligne 365. Il reprendra à la ligne suivante si vous faites :
CONT [RETURN]

```
310 REM STARTER 3
315 HIRES
320 P=0:I=1
330 CURSET 120,100,0
340 FOR R=1 TO 95 STEP 2
350 CIRCLE R,1
360 INK I:PAPER P
370 IF I<7 THEN I=I+1 ELSE I=1
380 IF P<7 THEN P=P+1 ELSE P=1
390 NEXT R
395 GOTO 320
```

Remarquez le END à la fin du programme. Il clôt le programme. Mais si vous l'enlevez (tapez 4ØØ, puis [RETURN] pour effacer la ligne), à l'exécution, vous constatez que le programme s'arrête. END est facultatif, dans ce cas. Nous verrons, plus loin, qu'il a son importance.

Starter numéro 4

```
339 REM STARTER 4
340 FOR R=1 TO 95
345 GOSUB 400
390 NEXT R
395 GOTO 340
400 SOUND 1,500-5*R,5+R/10
402 SOUND 2,1000-10*R,5+R/10
403 SOUND 3,2000-9*R,5+R/10
410 PLAY 7,0,1,2000
420 RETURN
```

L'utilisation d'une imprimante

Après branchement et connection de l'imprimante, si votre programme est en mémoire vive, tapez :
LLIST
Celui-ci sera listé sur l'imprimante au lieu de l'écran. Si vous n'obtenez pas de démarrage ou si les caractères sont indéchiffrables, reportez-vous à la fiche de référence : commandes de l'imprimante.

Note : Si vous possédez la cassette qui peut accompagner ce livre vous n'avez pas besoin de taper les 4 programmes "starter". Ils sont dans le premier enregistrement de cette cassette. Lisez la notice d'utilisation, regardez bien la liste d'instructions, et bon voyage !

5... Et c'est parti !

Tout effort trouve sa récompense : voici une mise en séquence de nos quatre starters qui vous permettra d'obtenir votre premier "gros" programme ! Avec quelques "modifs" pour varier les plaisirs !



Ce programme est le premier de la cassette. Lorsque le titre du programme est entouré de + c'est qu'il figure sur celle-ci.

```
1 REM+++SUPERSTART+++
2 REM
5 CLS
30 PAPER2
40 INK1
50 PRINT"QUELLE EST LA COULEUR";
51 PRINT" DES CARACTERES"
55 INPUT R$
60 IF R$<>"ROUGE"THEN 50
70 PRINT"BRAVO VOUS N'ETES PAS DALTONIEN"
80 WAIT 300
90 PRINT
100 PRINT"DONNEZ DEUX NOMBRES";
101 PRINT" SEPARES PAR UNE VIRGULE"
102 INPUT X,Y
110 PRINT
120 PRINT"LEUR SOMME EST ";X+Y
130 PRINT"LEUR PRODUIT EST ";X*Y
140 PRINT"LEUR QUOTIENT EST ";X/Y
141 PRINT:PRINT
143 PRINT"SI VOUS ETES D'ACCORD";
144 PRINT" AVEC LES RESULTATS,";
145 PRINT" APPUYEZ SUR UNE TOUCHE"
146 GET A$
147 CLS
150 PAPER4
160 A$="BONJOUR,MON NOM EST ORIC ET VOUS"
170 PRINT A$
180 INPUT NOM$
190 PRINT"BIENVENUE A BORD "NOM$
200 WAIT 250
210 PRINT
220 PRINT"VOTRE NOM A "LEN(NOM$);
221 PRINT" CARACTERES"
225 PRINT
230 PRINT" IL COMMENCE PAR UN: "LEFT$(NOM$,1)
235 PRINT
240 PRINT" ET SE TERMINE PAR UN: "RIGHT$(NOM$,1)
250 PRINT:PRINT"ATTENTION AU DECOLLAGE!!"
300 WAIT 600
315 HIRES
320 P=0:I=1
330 CURSET 120,100,0
340 FOR R=1 TO 95 STEP 2
350 CIRCLE R,1
360 INK I:PAPER P
365 GOSUB 400
370 IF I<7 THEN I=I+1 ELSE I=1
380 IF P<7 THEN P=P+1 ELSE P=1
385 NEXT R
386 ZAP:ZAP:ZAP:WAIT 100:TEXT:PAPER 0:INK 1
390 EXPLODE:WAIT 200
391 PLOT 2,7,14:PLOT 3,7,10
392 PLOT 4,7,"BIENVENUE DANS L'HYPER-ESPACE!"
```

```
393 PLOT 2,8,14:PLOT 3,8,10
394 PLOT 4,8,"BIENVENUE DANS L'HYPER-ESPACE!"
398 PRINT CHR$(19)
399 END
400 SOUND 1,500-5*R,5+R/10
402 SOUND 2,1000-10*R,5+R/10
403 SOUND 3,2000-9*R,5+R/10
410 PLAY 7,0,1,2000
420 RETURN
```

Attention : en fin de programme vous êtes dans l'hyper-espace. Pour revenir sur terre et dialoguer avec ATMOS, retournez-le et, avec la pointe d'un stylo, appuyez sur la touche située dans la fenêtre. Cette touche [RESET] ramène le fonctionnement normal sans perdre le programme.
Un dernier conseil : comme ce programme est plus long que la hauteur de l'écran, vous pouvez l'arrêter en cours de listing en appuyant une fois sur la barre d'espacement :
LIST
[ESPACEMENT] (une fois, pour arrêter)
[ESPACEMENT] (une fois, pour repartir)

CHECK-LIST

Tout pilote, avant le décollage, vérifie le bon fonctionnement de ses commandes à l'aide d'une check-list ou liste de pointage. Vous trouverez ici les mots-clé que vous avez appris dans ce chapitre. Les numéros de page renvoient aux fiches techniques où vous trouverez souvent des compléments d'information et des exemples.

[ESPACEMENT]	RUN 158	INK 201	CSAVE
[RETURN]	PRINT 162	ZAP 208	CLOAD
[SHIFT]	END 185	EXPLODE 209	LIST
[DEL]	SYNTAX ERROR	SHOOT 208	STOP
[CTRL]	Ready	RESET	CONT
[ESC]	PAPER 201	CLS	LLIST

Voici quelques suggestions de manipulations à effectuer pour appliquer les nouvelles acquisitions du chapitre :
— Effacer l'écran et renvoyer le curseur en haut et à gauche.
— Activer la sonnette.
— Rentrer une phrase du type : "BONJOUR, MON NOM EST ATMOS ET VOUS"

Décollage BASIC

Instructions pour le calcul

Attention, ATMOS vous envoie un message !

Vous pouvez "parler" à ATMOS par l'intermédiaire du clavier. Souvenez-vous des programmes starter.
Mais ATMOS doit aussi pouvoir vous communiquer ses résultats. Relisons le STARTER1.

```
...
100  INPUT "DONNEZ DEUX NOMBRES SÉPARÉS PAR UNE
     VIRGULE" ;X,Y
120  PRINT "LEUR SOMME EST : ";X+Y
130  PRINT "LEUR PRODUIT EST: ";X*Y
140  PRINT "LEUR QUOTIENT EST: ";X/Y
```

PRINT est l'instruction par laquelle ATMOS écrit sur l'écran les résultats de sa réflexion. Ce mot signifie imprimer en anglais. Cela vient des premiers ordinateurs qui imprimaient tous les résultats sur papier. Avec l'écran vidéo c'est un peu rétro, mais qu'importe.
Essayons. Tapez :
PRINT CALCUL
0
Ready
Cette machine est stupide ! Non, nous nous sommes livrés à un test — sans méchanceté — pour vérifier vos réflexes !
Évidemment, vous savez depuis votre "embarquement sur ATMOS" que pour faire afficher un message, il faut mettre le texte entre guillemets.
PRINT "CALCUL"
CALCUL
Ready
Cette machine est extraordinaire !
En informatique on appelle les phrases des chaînes de caractères. Nous y reviendrons dans le chapitre suivant. Pour l'instant, nous en savons suffisamment.
Vous pouvez remplacer PRINT par [?] ça fait gagner du temps.
?20
20
Ready

Quelques petits calculs

Et les nombres ? Va-t-il comprendre ? Guillemets ou pas guillemets ?
Essayons :
PRINT 33 (comme chez le médecin)
33
Ready

On m'avait toujours dit que les ordinateurs ça calculait. Donc, ça comprend les nombres ! Rien de plus normal !
Ça doit même savoir compter :
PRINT 2*2 (attention, le signe × est remplacé par * et : par /)
4 (Bravo)
PRINT 3/2
1.5 (Très bien)
Et les guillemets alors ?
PRINT "33"
33 (c'est donc la même chose)
PRINT "2*2"
2*2

Vous avez là une différence fondamentale. Les nombres sans guillemets sont considérés comme des nombres et les signes comme des opérateurs et les opérations sont effectuées par PRINT.
Si vous mettez des guillemets, tout ce qui se trouve entre guillemets est considéré comme caractères sans signification particulière et l'instruction PRINT affiche tous les caractères.

Revenons à nos nombres :
Les opérations sont celles de l'arithmétique + − * / et ↑ (élévation à la puissance)

Vous pouvez combiner les opérations mais il y a une hiérarchie dans les opérateurs.

La Reine est la parenthèse qui commande tous les autres. Ensuite vient l'↑. Puis à égalité / et * et enfin à égalité + et −.
Exemple :
? 6*3/4
4.5

? 6/3*4 (la machine a fait 6/3 et ensuite ×4)
8

? 4 + 8/2↑3
5

Entraînez-vous et si vous n'êtes pas sûr, mettez des parenthèses.

Un premier programme

Dans ce qui précède, nous avons travaillé en mode direct. L'ordinateur effectue une opération ou une instruction. Ensuite il oublie. On aimerait bien :
— qu'il n'oublie pas,
— qu'il effectue une série d'ordres.
C'est ce que nous avons fait dans le programme de démonstration. Pour que l'ordinateur retienne un ordre, il faut mettre un numéro devant.
Exemple :

```
10  PRINT 4+8/2↑3 [RETURN]
```

Vous avez écrit une ligne de programme.
N'oubliez surtout pas le RETURN, sinon la ligne ne sera pas retenue. Et il ne se passe rien. Même plus de READY. Pourtant ATMOS est sur les starting blocks. Il faut lui dire de partir. RUN, et il affiche 5. L'opération a été exécutée. Et vous pouvez recommencer autant de fois que vous le voudrez. Vous pouvez en rajouter un 2e en ligne 20.
(Remarquez qu'après une instruction PRINT, le curseur revient à la ligne)
Vous vous demandez peut-être à quoi cela sert un ordinateur puisque, pour effectuer d'autres calculs, il faut retaper la ligne 10.
Vous avez parfaitement raison. Tous les nombres utilisés jusque là étaient des constantes. L'ordinateur peut faire beaucoup mieux. Mais il faut, alors, utiliser des variables.

Les variables

Que ce mot ne vous effraie pas. C'est très simple. Prenons un exemple : vous voulez calculer le périmètre et la surface de cercles de différents rayons.
Le rayon est variable. On l'appellera RAYON, tout simplement. (Ou R si vous êtes fatigué). C'est une variable BASIC. Cette variable peut prendre toutes les valeurs numériques que vous voudrez lui donner.
Essayons :
PRINT R
0
Si vous n'avez aucune indication, la variable est automatiquement mise à zéro.
Alors comment lui donner une valeur ? Vous avez plusieurs possibilités :
• La première consiste dans le déroulement du programme à vous demander la valeur, vous la donnez et le programme repart. C'est le rôle de l'instruction INPUT.
Par exemple, reprenons le problème du cercle.

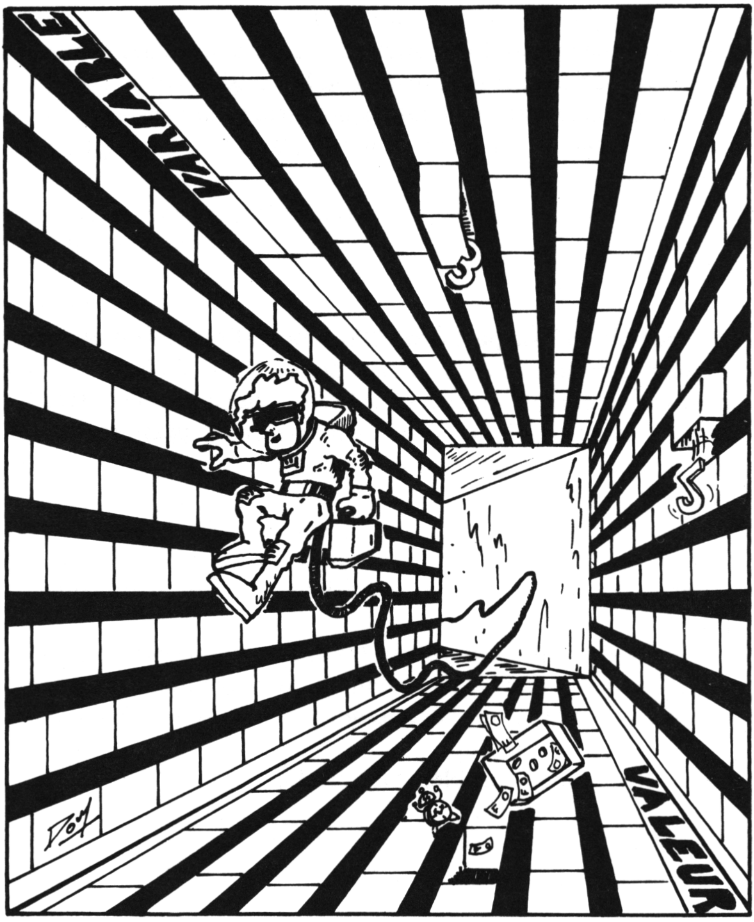
```
10  INPUT R
```

Cette instruction signifie : je te demande une valeur. Donne-la moi et je la mets dans la variable R.

```
20  PRINT R
30  PRINT 2*PI*R
40  PRINT PI×R↑2
    END
```

Que se passe-t-il à l'exécution ?
RUN

Vous voyez apparaître '?'
Vous tapez 3 [RETURN]
Et l'ordinateur affiche alors
3
18.8495559
28.2743339
Il a mis 3 dans R et a effectué le calcul. Vous avez remarqué au passage que ATMOS connaît p avec 8 décimales.
Ce programme est un peu sec. On ne sait pas ce que l'on demande ni ce que l'on affiche.



Heureusement INPUT et PRINT permettent de mélanger texte et résultats.
Vous écrivez alors :
10 INPUT "QUEL EST LE RAYON DU CERCLE"; R
Entre le message et la variable, n'oubliez pas le point virgule.
20 PRINT "POUR UN RAYON DE";R;"m, la circonférence est égale à"; 2*PI*R; "m"
30 PRINT "ET SA SURFACE VAUT"; PI*R↑2;"m2"
Quelques explications pour cette ligne : après un PRINT, vous pouvez avoir soit :
— un message entre guillemets. Il y a alors affichage du message. ("Pour un rayon de").
— une variable. L'ordinateur affiche la valeur de la variable (ici R).
— une expression arithmétique. L'ordinateur affiche le résultat. (Ici PI*R↑2).
Vous pouvez après un même PRINT faire plusieurs affichages sur la même ligne, juxtaposés. Il suffit de les séparer par des ; (c'est le cas ici).
Exécutez. Que se passe-t-il ?
QUEL EST LE RAYON DU CERCLE ? Vous tapez 3.
POUR UN RAYON DE 3 M, LA CIRCONFÉRENCE EST ÉGALE À 18.84...M ET SA SURFACE VAUT 28.27...M².
Parfait. Cet ordinateur parle comme un homme. Nous verrons, au chapitre suivant, comment il peut comprendre vos messages.
• La deuxième méthode pour donner une valeur à une variable est l'*affectation.*
Cela signifie que l'on donne à une variable une valeur.
Cela s'écrit : LET R = 3 qui signifie que R devienne égal à 3.
Par exemple dans le programme précédent, vous pouvez écrire :
10 LET R = 3
Et vous obtiendrez le même résultat.
ATMOS étant un ordinateur de la nouvelle génération, LET est facultatif.
Vous pouvez remplacer la ligne 10 par
10 R = 3
L'affectation est très souvent utilisée dans les programmes.
— D'une part, cela permet de fixer une variable au début du programme. Cela s'appelle l'initialisation.
— D'autre part, vous pouvez affecter à une variable une expression mathématique.
Dans l'exemple précédent, vous auriez pu écrire :

```
10  INPUT "QUEL EST LE RAYON DU CERCLE";R
15  P = 2*PI*R
16  S = PI*R↑2
40  PRINT R;P;S
```

Attention, lorsque vous affectez une variable à une autre, celle de gauche reçoit celle de droite.
Exemple : 50 X = Y
signifie X reçoit Y ou X prend la valeur de Y. Attention, Y n'est pas modifié.
C'est un peu l'inconvénient d'avoir rendu LET falcutatif. Le signe égal de

l'affectation n'est pas symétrique. (Certains langages utilisent un autre signe, par ex. ← ou : =)
Essayez par exemple :

```
10  X=5 : Y=10
20  X=Y
30  Y=X
40  PRINT X; Y
```

Auriez-vous prévu le résultat ?

Et on recommence

Vous avez calculé le périmètre et la surface pour R = 3. Vous aurez certainement besoin de le calculer pour un autre rayon.
Évidemment vous pouvez faire RUN. Mais c'est un peu fastidieux. Il vaudrait mieux revenir automatiquement à la ligne 10.
Vous le ferez avec l'instruction GOTO.
GOTO détourne le déroulement normal du programme et la ligne indiquée.



Ici, ajoutez :

```
50 GOTO 10
```

Le programme revient à la ligne 10 et vous repose la question. Vous entrez une nouvelle valeur et il vous affiche les nouveaux résultats.
Après quelques essais, vous serez peut-être lassé. Il faut arrêter cette machine. Comment faire ?
Essayons :

STOP
REDO FROM START ?

Ça signifie quoi ? Tout simplement que l'ordre INPUT suivi d'une variable vous demande un nombre. Vous lui donnez un mot, et il vous dit de recommencer.
Alors comment faire ? Vous avez oublié la 1re leçon. Appuyez sur CTRL C. Le programme est arrêté.
Vous avez vu l'intérêt du GOTO. Vous pouvez aiguiller le programme vers un autre endroit. Cela s'appelle une *rupture de séquence.* Effacez le précédent et essayez ceci :

```
10 GOTO 10
```

Il ne se passe rien mais vous avez perdu le contrôle du clavier. Le programme (d'une seule ligne) tournera indéfiniment tant que vous ne l'arrêterez pas par CTRL C. Vous verrez que c'est utile dans certains programmes graphiques pour garder un écran propre.
En voici, un autre :

```
10 ? "ATMOS";
20 GOTO 10
```

L'écran se couvre d'ATMOS. Vous pouvez, en ajoutant plus ou moins de blancs dans les guillemets, avoir des ATMOS alignés ou décalés.
Vous avez pu constater que l'instruction GOTO est simple et puissante. Elle vous permet un "pilotage" très précis. N'en abusez pas cependant. Elle peut rendre, si on l'utilise trop, les programmes très confus.
Avant de pousser à des choses un peu plus difficiles, une petite remarque qui vous fera gagner du temps. Vous pouvez mettre plusieurs instructions par ligne. Il suffit de les séparer par deux points (:).
Par exemple, si vous reprenez le 2e programme du cercle, vous pouvez écrire :

```
10 INPUT "QUEL EST LE RAYON DU CERCLE";R
20 P=2*PI*R : S=PI*R↑2
30 PRINT "R= "; R; "P=";P; "S=";S
40 GOTO 10
```

---

(1) ?PI : 3.14159265

Distinguez bien les [;], séparateurs dans l'instruction PRINT des [:] séparateurs d'instructions.
Vous obtiendrez ainsi des programmes plus condensés. C'est parfois commode pour avoir un listing qui prend moins de place.
Par contre, si vous faites des erreurs, vous aurez plus de mal à les trouver du fait qu'ATMOS indique uniquement la ligne de l'erreur.

Dans la vie, il faut faire des choix...

L'inconvénient du programme précédent est qu'il ne s'arrête jamais (sauf par CTRL C).
Il serait plus commode de convenir d'un code qui entraînerait l'arrêt du programme. Par exemple lorsque l'ordinateur vous interroge, si vous lui donnez zéro comme rayon, il s'arrête. (Il est bien évident qu'un cercle de rayon zéro a pour circonférence zéro. Donc le calcul, dans ce cas, ne servirait à rien).
Il faut alors qu'ATMOS puisse tester si R=Ø et, dans ce cas, il arrête l'exécution.
C'est très simple : vous utilisez l'instruction IF condition THEN instruction qui signifie : si la condition est remplie alors j'exécute l'instruction ou les instructions après THEN, dans le cas contraire, je passe à la ligne suivante.
Allons-y : il faut introduire la condition avant le GOTO, sinon le programme n'y passerait jamais. Nous écrirons alors :

```
35  IF R=Ø THEN END
```

la suite restant la même, à savoir :

```
4Ø  GOTO 1Ø
```

Essayez :
Vous faites vos calculs normalement, tant que R≠Ø, le programme passe à 4Ø donc à 1Ø. Quand vous avez fini, vous tapez Ø [RETURN] et le programme s'arrête puisque END entraîne une interruption de l'exécution.
Remarquez que vous auriez pu écrire le programme de la façon suivante :
Si R est différent de Ø recommencer. Le signe mathématique ≠ s'écrit en BASIC <>.

```
35  IF R<>Ø THEN 1Ø
4Ø  END
```

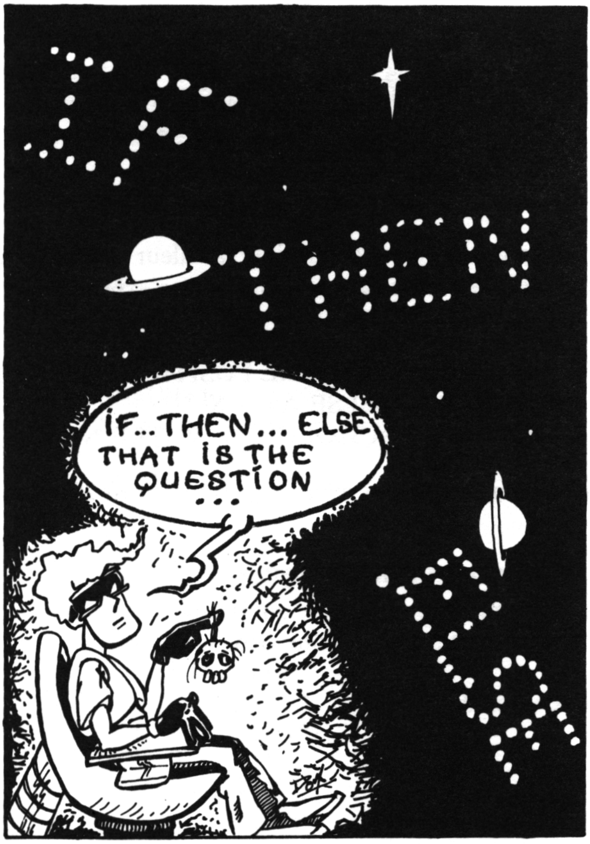
(Après le THEN, ce n'est pas une instruction mais un n° de ligne. Cette notation équivaut à GOTO 1Ø) : si R≠Ø alors on revient au début. Sinon, on s'arrête.
Une utilisation très fréquente du IF est la suivante. Très souvent, dans un INPUT vous ne pouvez pas accepter certaines valeurs.
Par exemple, dans ce programme, un rayon négatif ne signifie rien. Cela donnerait un périmètre négatif. (Mais dans certains cas, cela peut conduire à une erreur dans une fonction d'où un arrêt du programme).

Il faut donc prévenir la personne qui utilise le programme ou au moins lui reposer la question.
Cela donne le programme suivant :

```
5 REM RAYON
6 REM
7 CLS
10 INPUT "QUEL EST LE RAYON DU CERCLE ";R
20 P=2*PI*R :S=PI*R^2
25 IF R<0 THEN PRINT ATTENTION R<0":GOTO 10
30 PRINT "R= ";R;"P= ";P;"S= ";S
35 IF R<>0 THEN 10
40 END
```

Essayez maintenant d'introduire un rayon négatif : on vous dit que ça ne convient pas et on vous prie poliment de recommencer.
Une ligne de ce type s'appelle un filtrage ou une protection. C'est presque toujours nécessaire (avec ou sans message) après un INPUT pour un programme sérieux.

Et on choisit souvent entre deux solutions



Nous venons de voir l'instruction IF... THEN... qui permet, si une condition est réalisée de faire quelque chose et dans le cas contraire de continuer le programme normalement.
C'est un peu comme un menu au restaurant avec fromage et dessert. On pourrait dire :
Si vous aimez ça, alors prenez du fromage
Dessert
Il y a des gens qui n'aiment pas le fromage mais ils auront de toute façon du dessert.
Mais de plus en plus, en ces temps difficiles, les restaurants proposent fromage ou dessert.
Cela signifie pour vous :
Si je préfère le fromage, alors je prends le fromage, sinon je prends le dessert.
SI condition ALORS instruction SINON instruction

En BASIC
IF condition THEN instruction ELSE instruction

Les deux instructions après THEN et ELSE sont exclusives, contrairement au cas précédent.
Un exemple : les nombres sont positifs ou négatifs : alors vous allez donner un nombre à ATMOS et il vous dira si le nombre est positif ou négatif

```
10  INPUT "DONNEZ UN NOMBRE POSITIF OU NÉGATIF";X
20  IF X>=0 THEN PRINT "LE NOMBRE EST POSITIF" ELSE
    PRINT "LE NOMBRE EST NÉGATIF"
30  END
```

Vous pouvez de la même façon déterminer la valeur absolue d'un nombre. La valeur absolue c'est la valeur sans le signe pour s'exprimer simplement. Cela donne ceci :

```
10  INPUT "DONNEZ UN NOMBRE POSITIF OU NÉGATIF";X
20  IF X>=0 THEN A=X ELSE A=-X
30  PRINT "LA VALEUR ABSOLUE DE VOTRE NOMBRE EST";A
```

C'est évident mais notez quand même l'affectation en 20. A reçoit X ou −X suivant les cas.

Quelques fonctions mathématiques

Nous venons de faire un programme qui donne la valeur absolue d'un nombre. Eh bien, vous avez travaillé pour rien. Cette fonction existe et s'appelle ABS.
Tapez en mode immédiat :

```
? ABS (-3)
3
? ABS (7)
7
```

ATMOS est vraiment très savant. Et il fait bien mieux encore. Il donne :
— La partie entière d'un nombre : INT

? INT (2.5)

2

? INT (7.9999)

7

Vous avez compris.

— La racine carrée : SQR

? SQR (2)

1.414

— Un nombre aléatoire : RND

Vous savez, le pile ou le face, les nombres d'un Dé. Vous ne pouvez pas savoir à l'avance ce qui va sortir.

ATMOS vous donne un dé électronique.

C'est la fonction RND.

Pour l'utilisation que nous en ferons RND (1) nous suffira et il donnera un nombre aléatoire compris entre Ø et 1 (1 exclu).

Si vous voulez simuler un dé, (nombres 1 à 6) il vous faudra écrire :

X = INT (RND(1)*6) + 1

INT (RND(1)*6) est un nombre de Ø à 5

Donc X sera un nombre compris entre 1 et 6.

Cela vous intéresse de voir si ce nombre est vraiment aléatoire ?

Avec un dé, c'est très long. Il faut lancer le dé et noter à chaque fois le résultat. Ensuite compter chaque chiffre et voir si les nombres sont voisins. ATMOS se chargera de cette besogne :

```
 10 REM COUP DE DE 1
 11 REM
 12 CLS
 15 PRINT"PATIENCE..."
 20 X=INT(RND(1)*6)+1
 25 PRINTX;
 30 N=N+1
 40 IF X=1 THEN UN=UN+1
 50 IF X=2 THEN DEUX=DEUX+1
 60 IF X=3 THEN TROIS=TROIS+1
 70 IF X=4 THEN QUATRE=QUATRE+1
 80 IF X=5 THEN CINQ=CINQ+1
 90 IF X=6 THEN SIX=SIX+1
100 IF N<=1000 THEN 20
105 PING
106 PRINT
110 PRINT"NOMBRE DE 1 ";UN
120 PRINT"NOMBRE DE 2 ";DEUX
130 PRINT"NOMBRE DE 3 ";TROIS
140 PRINT"NOMBRE DE 4 ";QUATRE
150 PRINT"NOMBRE DE 5 ";CINQ
160 PRINT"NOMBRE DE 6 ";SIX
170 END
```

Cela amène quelques explications :
en 12, CLS est l'instruction d'effacement d'écran
en 2Ø, vous avez le nombre aléatoire compris entre 1 et 6
en 3Ø, vous avez réalisé un *compteur*. Vous reconnaissez le signe de l'affectation.

Remarquez que la variable est la même de part et d'autre N = N + 1 signifie que N reçoit l'ancienne valeur augmentée de 1. On parle aussi d'*incrémentation*.

C'est très courant en informatique. Et ça sert à quoi ?

Sautez à la ligne 1ØØ. Si N < 1ØØØ, on recommence. Sinon on affiche les résultats, c'est fini. Cela signifie que vous allez lancer le dé 1ØØØ fois et qu'après vous allez regarder les résultats.

Enfin les lignes 4Ø à 9Ø sont également des compteurs mais des compteurs partiels.

Si X = 1 le compteur de un est augmenté de 1 et ainsi de suite.

Et vous pouvez ainsi tester votre fonction RND. Et si vous êtes courageux et si vous avez le temps vous pouvez toujours faire 1ØØØØ ou 1ØØØØØ essais. Puisque ATMOS travaille tout seul, vous pouvez faire autre chose en attendant.

Attention, toutes ces fonctions donnent un résultat. Ce résultat doit être intégré à une instruction ou une expression mathématique.

Par exemple :

? SQR(7) — affichage

ou X = SQR(7) — affectation

1Ø SQR(7) ne signifie rien en BASIC et vous donnera une erreur de syntaxe.

La capacité de calcul d'ATMOS

Les variables numériques qu'utilise ATMOS sont de deux types : simple précision, entier (accompagnées de l'indice %).

ORIC ATMOS peut traiter des valeurs entières entre 32767 et −32768

Le nombre le plus important qu'ATMOS peut traiter est 1.7Ø141E+38, le plus petit est 2.93874E−39.

Ces deux nombres sont exprimés en exponentielle ou notation scientifique que l'ordinateur utilise à partir du rang 999999999 et Ø.Ø1. Ce système permet d'augmenter la puissance de calcul tout en économisant la place prise par les nombres en mémoire.

Il définit de combien la valeur décimal avant E doit être multipliée (E+) ou divisée (E−)

Ces points sont repris et développés dans les chapitres consacrés à la mémoire et au langage machine. En attendant, voici deux programmes qui montrent les capacités de calcul du plus grand au plus petit nombre et comment sont notés les résultats.

```
5 REM LE PLUS GRAND NOMBRE
10 N=2
20 N=N*8
30 PRINTN
40 WAIT 10:GOTO20
```

```
5 REM LE PLUS PETIT NOMBRE
10 N=3
20 N=N/2
30 PRINT N
40 WAIT 10 :GOTO 20
```

```
16
128
1024
8192
65536
524288
4194304
33554432
268435456
2.14748365E+09
1.71798692E+10
1.37438954E+11
1.09951163E+12
8.79609302E+12
7.03687442E+13
5.62949954E+14
4.50359963E+15
```

Boucles

On a souvent besoin de répéter un calcul un certain nombre de fois. Nous avons un exemple avec les dés. On utilise une variable qui augmente de 1 à chaque fois. Et lorsque le nombre voulu est atteint, on arrête la répétition par un IF.
Vous savez bien qu'ATMOS ne cherche qu'à vous rendre service. Il fera ce travail à votre place avec l'instruction FOR...NEXT.
Dans l'exemple précédent, vous écrirez :

10 FOR N = 1 TO 1000
20 X = INT (RND(1)*6) + 1
30 ligne supprimée.
40
à lignes inchangées
90
100 NEXT N (équivaut au test de sortie du programme précédent).

110
à lignes inchangées
160

Ce programme fonctionne exactement de la même façon que le précédent. Il est plus simple à écrire et plus simple à comprendre.
Vous savez que entre le FOR N et le NEXT N, vous avez une boucle qui se répétera 1000 fois. Cette instruction est très souvent utilisée. Vous pouvez la combiner avec des instructions PRINT pour afficher plusieurs fois ou avec des INPUT pour demander un certain nombre de valeurs.

Voici quelques exemples :

```
10 REM+++COUP DE DE 2+++
11 REM
12 CLS
13 PRINT"PATIENCE..."
14 FOR N=1 TO 1000
20 X=INT(RND(1)*6)+1
25 PRINT X;
40 IF X=1 THEN UN=UN+1
50 IF X=2 THEN DEUX=DEUX+1
60 IF X=3 THEN TROIS=TROIS+1
70 IF X=4 THEN QUATRE=QUATRE+1
80 IF X=5 THEN CINQ=CINQ+1
90 IF X=6 THEN SIX=SIX+1
100 NEXT N
105 PING
106 PRINT
110 PRINT"NOMBRE DE 1 ";UN
120 PRINT"NOMBRE DE 2 ";DEUX
125 PRINT"NOMBRE DE 3 ";TROIS
130 PRINT"NOMBRE DE 4 ";QUATRE
150 PRINT"NOMBRE DE 5 ";CINQ
160 PRINT"NOMBRE DE 6 ";SIX
162 WAIT 300
195 PING
164 PRINT:INK5:PRINT"Un coup de de,JAMAIS "
165 PRINT"n'abolira le hasard."
166 PRINT TAB(20)"MALLARME"
170 END
```

```
10 REM TABLEAU DES CARRES CUBES ET RACINES
15 CLS
20 PRINT"NOMBRE","CARRE","CUBE","RACINE"
25 PRINT
30 FOR X=1 TO 20
40 PRINTX,X*X,X*X*X,SQR(X)
50 NEXT X
60 END
```

Et ATMOS vous affichera le tableau pour les 20 premiers entiers. Mais vous aurez peut-être besoin des valeurs pour les nombres pairs uniquement. L'instruction FOR… NEXT comporte un terme supplémentaire facultatif : STEP
Remplacez la ligne 30 par
30 FOR X=2 TO 40 STEP 2
Le reste est inchangé. Vous aurez le tableau pour les nombres pairs.
STEP (PAS) commande l'augmentation de la variable de la boucle. S'il est omis, la variable augmente de 1. Vous pourriez aussi bien mettre STEP 1. Si vous précisez un autre pas P, à chaque boucle, la variable est augmentée de P. Ici elle était augmentée de 2.

Remarquons que P peut être négatif mais attention alors aux limites de la variable qui est décroissante.
Ainsi, vous pouvez écrire, en 3Ø toujours :
3Ø FOR X=4Ø TO 2 STEP −2
et vous aurez le même tableau mais par valeurs décroissantes.
Sur les ordinateurs "ordinaires", la boucle FOR NEXT est souvent utilisée pour réaliser une temporisation c'est-à-dire une attente.

Par exemple :

```
2Ø  CLS
3Ø  PRINT "RÉFLÉCHISSEZ PENDANT QUELQUES SECONDES"
4Ø  FOR T=1 TO 3ØØØ : NEXT
5Ø  CLS
6Ø  REM SUITE DU PROGRAMME
```

En ligne 4Ø, l'ordinateur compte et ne fait rien d'autre jusqu'à 3ØØØ. Ensuite, il continue le programme.
Mais vous avez bien compris que ATMOS est un ordinateur performant. Vous pouvez remplacer la ligne 4Ø par
4Ø WAIT 3ØØ et l'ordinateur attendra 3 secondes. WAIT N signifie qu'il attend pendant N/1ØØ secondes.
Vous verrez, c'est très utile dans les jeux et pour la musique.
Un autre exemple :

```
1Ø  CLS
5Ø  FOR I=1 TO 38  }
6Ø  PRINT "Ø";     } boucle ligne
7Ø  NEXT I         }
```

Et vous avez une ligne de Ø.
Vous pouvez insérer cette boucle dans une autre. Supposons que les lignes 5Ø,6Ø,7Ø soient une seule et même instruction qui remplisse une ligne de Ø.
Si nous voulons maintenant remplir l'écran qu'allons-nous faire ?
Nous écrirons :

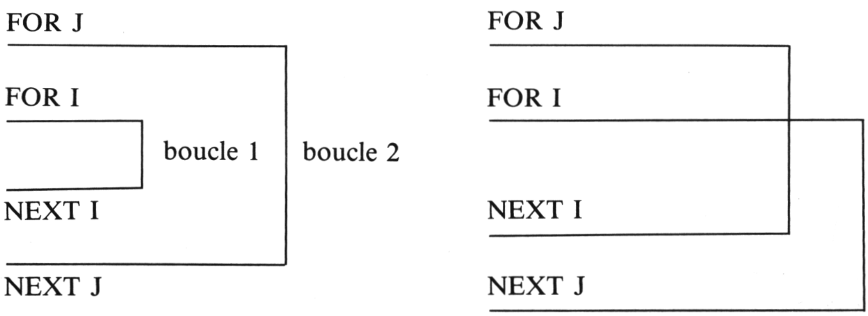
```
4Ø  FOR J=1 TO 26                 (26 lignes d'écran)
5Ø
6Ø  Boucle ligne : remplit une ligne
7Ø
8Ø  PRINT               On passe à la ligne suivante
9Ø  NEXT J              On recommence
```

Si vous écrivez le programme complet :

```
2Ø  CLS
4Ø  FOR J=1 TO 26
5Ø  FOR I=1 TO 38
6Ø  PRINT "Ø";
7Ø  NEXT I
8Ø  PRINT
9Ø  NEXT J
```

vous avez deux *boucles imbriquées.* Pour la clarté du programme, nous avons décodé la deuxième boucle.
Vous pouvez constater que les variables des NEXT sont dans l'ordre inverse de celles des FOR. C'est une règle très générale pour l'utilisation des boucles.

Vous aurez toujours ceci et jamais cela



Vous verrez dans la suite que les boucles sont systématiquement utilisées toutes les fois que l'on veut répéter un nombre fini de fois.
Prenez-en l'habitude et entraînez-vous.
Pour vous y aider voici un petit jeu d'anagramme qui fait la synthèse de ce que vous venez d'apprendre et la liaison avec le chapitre suivant.
Remarquez l'ordre des FOR et des NEXT, certaines parties du programme présentent des particularités que vous connaîtrez plus loin.

```
10 RM+++ANAGRAMME+++
11 REM
20 CLS
50 FOR I=1 TO 5
60 PRINT"LETTRE: ";I;
70 INPUT L$(I)
80 NEXT I
90 PRINT
95 PAPER3:INK4
100 FOR I=1 TO 5
110 FOR J=1 TO 5
120 IF I=J THEN 200
125 FOR K=1 TO 5
130 IF I=K OR J=K THEN 190
140 FOR L=1 TO 5
145 IF I=L OR J=L OR K=L THEN 180
165 FOR M=1 TO 5
170 IF I=M OR J=M OR K=M OR L=M THEN 175
175 PRINT L$(I);L$(J);L$(K);L$(L);L$(M);"-";
178 NEXT M
180 NEXT L
190 NEXT K
200 NEXT J
210 NEXT I
215 SHOOT
220 END
```

```
LETTRE: 1 F
LETTRE: 2 A
LETTRE: 3 K
LETTRE: 4 I
LETTRE: 5 R
FAKIF-FAKIA-FAKIK-FAKII-FAKIR-FAKRF-FAKRA-FAKRK-FAKRI-FAKRR-FAIKF-F
AIKA-FAIKK-FAIKI-FAIKR-FAIRF-FAIRA-FAIRK-FAIRI-FAIRR-FARKF-FARKA-FA
RKK-FARKI-FARKR-FARIF-FARIA-FARIK-FARII-FARIR-FKAIF-FKAIA-FKAIK-FKA
II-FKAIR-FKARF-FKARA-FKARK-FKARI-FKARR-FKIAF-FKIAA-FKIAK-FKIAI-FKIA
R-FKIRF-FKIRA-FKIRK-FKIRI-FKIRR-FKRAF-FKRAA-FKRAK-FKRAI-FKRAR-FKRIF
-FKRIA-FKRIK-FKRII-FKRIR-FIAKF-FIAKA-FIAKK-FIAKI-FIAKR-FIARF-FIARA-
FIARK-FIARI-FIARR-FIKAF-FIKAA-FIKAK-FIKAI-FIKAR-FIKRF-FIKRA-FIKRK-F
```

CHECK-LIST

PRINT 162
Variable numérique
REDO FROM START
GOTO X 187
INPUT 161
LET 169
+ = > < <>
IF... THEN... 188
IF... THEN... ELSE 188
ABS 174
INT 176
RND 175
SQR 171
FOR... NEXT 189
STEP 189
WAIT N 187
PI

TRAINING

• Écrire un programme calculant la surface d'un carré, d'un rectangle, d'un triangle.
• Écrire un programme calculant depuis combien d'heures vous êtes nés. Voici la fonction du calcul : (365*ANS + 30*MOIS + JOURS)*24
• Écrire un programme qui calcule la factorielle d'un nombre.

Instructions pour l’écriture

ATMOS, vous le savez ne se pilote pas qu’avec des chiffres : voici venu le moment de manipuler les mots, les textes qu’il peut emmagasiner dans sa mémoire.

Qu’est-ce qu’une chaîne de caractères ?

C’est tout ce qui est écrit entre guillemets après un PRINT ou un INPUT.
En termes plus précis, une chaîne de caractères est une suite de caractères qui peut comprendre :
— des caractères alphabétiques : A,B,C... Y,Z et des minuscules a,b,c... y,z

PRINT “ATMOS A UNE MÉMOIRE FABULEUSE”
PRINT “atmos a une mémoire fabuleuse”

— des chiffres : Ø, 1, 2, 3... 9
PRINT “48K DE MÉMOIRE VIVE”
Attention, le fait de placer “48” entre guillemets signifie qu’on ne considère pas ce chiffre comme une valeur numérique mais comme une chaîne de caractères.
— des signes de ponctuation : , ; . ! : ( ) ?
PRINT “ATMOS? FABULEUSE MÉMOIRE!”
— signes ou symboles particuliers : + – * / % < >
PRINT “48K + 16K = 64K”
— des caractères spéciaux : le retour en début de ligne, le passage à la ligne, etc.
— un caractère d’espacement appelé aussi “blanc” :
PRINT ” ”
Une suite de caractères entre guillemets s’appelle une *constante chaîne.* Sur une suite de caractères, comme sur un mot, il est possible d’opérer toutes sortes de traitements que nous allons décomposer ensemble.

Les variables chaînes

Écrire des chaînes entre guillemets est un exercice qui cesse d’être amusant lorsqu’il faut les répéter à chaque opération. Heureusement, le système des variables permet de s’en dispenser.
Ranger une constante chaîne dans une variable chaîne consiste tout simplement à lui donner un nom :
A$ = “BONJOUR, MES” revient à dire que la chaîne de caractères “BONJOUR, MES” sera désormais désignée, dans le programme, par la variable A$.
Autre exemple, si l’on veut que la variable B$ représente, une fois pour toutes, la chaîne “DAMES”, on écrira l’instruction comme suit :
B$ = “DAMES”

Et, de la même façon, on affectera à C$ la chaîne "SIEURS":
C$ = "SIEURS"
De cette façon, on pourra combiner les différentes variables pour souhaiter la bienvenue dans le programme que voici :

```
5 REM BONJOUR
10 CLS
20 A$="BONJOUR ,MES"
30 B$="DAMES"
40 C$="SIEURS"
50 PRINTA$;B$
60 PRINT
70 PRINTA$;C$
80 END
```

Vous l'avez, sans doute, noté au passage, les variables contenant des chaînes de caractères sont toujours suivies de $: A$, B$, C$...
Ce symbole permet de différencier ces variables de l'autre type de variables qui contiennent des valeurs numériques et qu'on appelle variables numériques comme vous l'avez vu dans le chapitre précédent.

Dialoguer avec ATMOS

Dans le programme précédent, ATMOS est seul à "parler". Voici comment entamer le dialogue avec lui :

10 CLS
20 A$ = "BONJOUR, MON NOM EST ATMOS ET TOI ?"
30 PRINT A$
40 INPUT NOM$ REM à ce moment, ATMOS attendra que vous écriviez la suite de caractères que comprend votre prénom. Il l'appellera NOM$, dès que vous taperez [RETURN]
50 PRINT "SUR CET ÉCRAN,"; NOM$; "J'écris ton nom !"
60 GOTO 50
70 END
RUN
(Pour arrêter [CTRL] [C])
(Pour nettoyer l'écran [CLS])

Il est souhaitable, naturellement, d'améliorer la communication entre êtres humains : ATMOS, lui, ne demande que cela ! Alors, comment s'y prendre ?
On peut afficher des questions et des réponses stéréotypées dans le style de celles du programme précédent.
On peut, également, nuancer les réactions de notre interlocuteur électronique en fonction de la réponse de l'utilisateur aux questions posées. Le moyen ?
Recourir à une instruction IF... THEN... ELSE... qui permet de tester la réponse de l'utilisateur et d'aiguiller ATMOS vers des réponses appropriées, prévues à l'avance.

Voici donc la suite du "dialogue" de bienvenue avec ATMOS :

```
10 CLS
20 A$="BONJOUR,MON NOM EST ORIC ET TOI ?"
30 PRINT A$
40 INPUT NOM$
50 PRINT "J'AI CONNU UN ";NOM$
60 PRINT"MAIS IL ETAIT MOINS RELAXE QUE TOI!"
70 INPUT "VEUX TU ALLER PLUS LOIN AVEC MOI";R$
80 IF R$="OUI" THEN C$="MON CHER ":GOTO 90
85 C$="DOMMAGE ":D$="AURIONS FAIT":GOTO 100
90 D$="ALLONS FAIRE"
100 FOR I=1 TO 12
110 FOR J=1 TO 12
120 PRINTC$;NOM$
130 PRINT
140 PRINT"NOUS ";D$;" DE"
150 PRINT"GRANDES ET BELLES CHOSES ENSEMBLE"
160 END
```

Épilogue au "dialogue"

Vous avez tout prévu dans ce dialogue, les questions comme les réponses. Mais si un autre interlocuteur d'ATMOS se mettait à répondre autre chose que ce qu'on attend de lui ?

Supposez, par exemple, qu'à la question du programme précédent :

70 INPUT "VOULEZ-VOUS ALLER PLUS LOIN AVEC MOI"; R$

l'interlocuteur réponde "D'ACCORD" ou bien "NON MERCI" au lieu du "OUI" ou du "NON" attendu à la ligne suivante ?

Faites à nouveau tourner le programme pour tenter l'expérience.

Si vous tapez "D'ACCORD", ATMOS en ligne 80 constate que R$ n'est pas égal à "OUI" en conséquence il exécute l'instruction qui suit ELSE ("tout autre réponse") ce qui revient à considérer que vous avez répondu "NON"!

Pour mettre fin à ce dialogue de sourd, il suffit de prévoir d'autres réponses, nous direz-vous.

La solution est plus simplette mais efficace : reposer la question si la réponse n'est pas celle qui est attendu. Pour cela, il suffit d'ajouter une ligne d'instruction :

75 IF R$< > "OUI" AND R$< > "NON" THEN 70

mot-à-mot : si la réponse est *différente* de "OUI" *et* si elle est *différente* de "NON" retourner en ligne 70.

Vous retrouvez ici, dans un contexte différent, la ligne de filtrage ou de *protection* nécessaire après un INPUT que vous aviez déjà vue dans le chapitre précédent.

```
5 REM+++DIALOGUE+++
6 REM
10 CLS
20 A$="BONJOUR,MON NOM EST ORIC ET TOI?"
30 PRINT A$
40 INPUT NOM$
50 PRINT"J'AI CONNU UN ";NOM$
```

```
60 PRINT "MAIS IL ETAIT MOINS RELAXE QUE TOI!"
70 INPUT"VEUX-TU ALLER PLUS LOIN AVEC MOI";R$
80 IF R$="OUI" THEN C$="MON CHER ":GOTO 90
85 C$="DOMMAGE ":D$="AURIONS FAIT":GOTO 100
90 D$="ALLONS FAIRE"
100 FOR I=1 TO 12
110 FOR J=1 TO 12
120 PRINT C$;NOM$
130 PRINT
140 PRINT"NOUS ";D$;" DE"
150 PRINT" GRANDES ET BELLES CHOSES ENSEMBLE"
160 END
```

Concaténer les mots ?

C'est tout simplement les mettre "bout à bout".
Pour quoi faire ?
Des phrases, bien sûr.

```
10  A$="ETRE" : B$="OU"
20  C$="NE" : D$="PAS"
30  B$= A$ + B$
40  B$= B$ + C$ + D$ + A$
50  PRINT B$
RUN
ÊTRE OU NE PAS ÊTRE
```

That is the question... La question ici est de savoir comment ATMOS relie les mots ensembles, comment il forme une phrase.
Vous avez certainement repéré en ligne 30 le responsable. Eh oui, le signe "+", le même signe que pour l'addition. L'addition des variables chaînes A$ et B$ revient à les écrire bout à bout. ATMOS est logique !
Si vous voulez faire plaisir à ces messieurs les informaticiens vous direz "concaténation" au lieu d'addition et, plus fort encore, "opérateur de concaténation" pour signe + !
Il y en a un qui se fend le clavier, c'est ATMOS !
Trêve de plaisanterie, vous avez compris qu'il est possible de faire des "opérations" sur les mots comme ça l'est sur les nombres. Pour vous distraire et enrichir votre vocabulaire scientifique, voici une autre application de la CONCANE... pardon CONCATENATION :

```
5 REM+++LEXIQUE+++
6 REM
10 CLS
20 A$="BI"
30 B$="POLY"
40 C$="CYCLE"
50 D$="NOME"
60 X$=B$+D$
70 Y$=B$+C$
80 Z$=A$+D$
90 T$=A$+C$
100 PRINT X$ , Y$
110 PRINT Z$ , T$
120 END
```

A vous d'imaginer des variations sur le même thème. Sinon rendez-vous en fin de chapitre

Esca MOT er

Un escamoteur est un manipulateur d'objets. Il les fait apparaître ou disparaître, augmenter ou rétrécir, se démultiplier, se métamorphoser. Les "objets" que manipule ATMOS, ce sont les mots : en ce sens on peut dire qu'il est un sacré ESCA "MOT" EUR !
Vous allez tout savoir de ses "trucs".
Tapons, tout d'abord la chaîne suivante :
X$ = "MY ATMOS IS FABULOUS"
Et essayons, en mode immédiat ;
? MID$ (X$,4,1)
O
Poursuivons :
? MID$ (X$,4,3)
ATMOS
Encore une fois :
? MID$ (X$,4,4)
ATMOS
Ce "truc", MID$, est une fonction-chaîne. Dans le dernier exemple, la fonction MID$ a extrait 4 caractères à partir du 4e caractère de la chaîne X$:
"MY ATMOS IS FABULOUS"
1er 2e 3e 4e
MID$ (X$, 4, 4)
Ne pas oublier que les espaces comptent comme un caractère.
A vous de jouer : comment extraire la *sous-chaîne* "FABULOUS" de la chaîne X$? (1)

Passons maintenant à l'un des tours faciles du maître : l'écriture verticale. Papier, crayon, s'il vous plaît pour découvrir la logique programmatique qui fera s'afficher la chaîne X$ verticalement sur l'écran : le 1er caractère sur la première ligne, le 2e sur la seconde...
Un petit tuyau, avant de commencer : lorsqu'ATMOS doit répéter plusieurs fois la même action, il est recommandé de placer celle-ci dans une boucle FOR... TO... NEXT comme vous le savez.

```
10 REM   VERTICAL
11 REM
15 CLS
20 X$="MY ORIC IS FABULOUS"
30 FOR I=1 TO 19
40 PRINT MID$(X$,I,1)
45 INK 5
50 NEXT I
60 END
```

---

(1) ?MID$(X$, 12,8)

RUN
M
Y

O
R
etc.

La chaîne X$ contient 19 caractères. Le programme exécute 19 fois de suite l'affichage du caractère numéro I de la chaîne X. Chaque instruction PRINT imprime le caractère et provoque un retour en début de ligne puis un saut de ligne.

Ce programme est adaptable à d'autres chaînes de caractères, à condition de compter le nombre total de caractères (y compris les espaces) et d'effectuer la modification de la sortie de boucle (ligne 3Ø du programme).

Toutefois, si vous souhaitez vous épargner la peine de compter tous les caractères d'une chaîne (surtout si elle est longue), la fonction-chaîne LEN s'en chargera à votre place.
Testez-la en mode immédiat :
?LEN (X$)
19
Autre exemple :
? LEN ("MICROPROCESSEUR")
15
Effectivement la chaîne MICROPROCESSEUR contient 15 caractères.
Revenons, à présent, à notre programme d'affichage vertical. Voyez-vous comment utiliser cette fonction pour la rendre applicable à n'importe quelle phrase ?

Examinons ensemble la donnée principale : il s'agit du nombre de caractères de la phrase. Ce nombre peut varier. Comment le connaître ? Par LEN, bien sûr. Comment le faire savoir à la machine, quelle que soit la longueur de la phrase ? En remplaçant le chiffre de sortie de boucle en ligne 3Ø par la fonction LEN :

```
3Ø  FOR I=1 TO LEN(X$)
```

Il vous reste maintenant à donner à l'utilisateur la possibilité d'entrer n'importe quelle phrase. Cette facilité vous est offerte par INPUT. Voici donc votre programme modifié :

```
1Ø  REM AFFICHAGE VERTICAL
15  CLS
2Ø  INPUT "ÉCRIVEZ UN MOT OU UNE PHRASE";X$
3Ø  FOR I = 1 TO LEN (X$)
4Ø  PRINT MID$ (X$,I,1)
5Ø  NEXT I
```

Veuillez noter, au passage, que la fonction LEN rend un nombre comme résultat alors que celui de la fonction MID$ est une chaîne. Il existe, en effet, deux catégories de fonctions applicables aux chaînes de caractères : celles qui ont des résultats numériques et celles qui ont des résultats alphabétiques. Ces dernières se distinguent par le caractère $.

Deux autres fonctions permettent d'extraire d'une chaîne une partie de celle-ci. Il s'agit de LEFT$ et de RIGHT$. Essayons-les en mode immédiat :

```
?LEFT$ ("MICROPROCESSEUR",2)
MI
?LEFT$ ("MICROPROCESSEUR",5)
MICRO
?LEFT$ ("MICROPROCESSEUR",15)
MICROPROCESSEUR
?RIGHT$ ("MICROPROCESSEUR",1)
R
?RIGHT$ ("MICROPROCESSEUR",10)
PROCESSEUR
?RIGHT$ ("MICROPROCESSEUR",15)
MICROPROCESSEUR
```

Le truc est simple : LEFT$ permet d'extraire une sous-chaîne de n caractère(s) à partir de la gauche, RIGHT$ extrait de la même façon mais à partir de la droite.

Application ? Toute recherche d'information qui se fonde sur l'exploration d'une chaîne de caractères.

Supposons, par exemple, que connaissant la date d'achat d'un produit, on veuille la date d'échéance pour une traite de 90 jours.

Appelons D$ la date d'achat (chiffres entre guillemets séparés par des /:13/08/1983)

```
10 REM ECHEANCE
11 REM
15 CLS
20 INPUT "DATE D'ACHAT ";D$
30 M$=MID$(D$,4,2)
40 M=VAL(M$):REM Conversion de la chaine
41 :           REM en valeur numerique.
```

Le recours à cette nouvelle fonction VAL (M$) — tranformation d'une chaîne numérique en sa valeur numérique — est nécessaire pour effectuer le calcul de l'échéance. Continuons :

```
50 PRINT"Mois de l'achat ";M
60 E=M+3 :REM Calcul de l'echeance
70 PRINT"Mois de l'echeance: ";E
80 PRINT
90 PRINT"Date exacte d'echeance: ";
95 PRINTLEFT$(D$,3);E;RIGHT$(D$,3)
Mois de l'achat 5
Mois de l'echeance: 8

Date exacte d'echeance: 2/8 /83
```

Des lettres aux chiffres

Nous venons de voir, dans le programme précédent, qu'il est parfois nécessaire d'exploiter une chaîne de caractères comme une variable numérique ou inversement d'exploiter une valeur numérique comme une variable chaîne. Nous avons utilisé VAL(X$) qui convertit une chaîne numérique en sa valeur numérique. Attention, le contenu de la chaîne doit correspondre à la forme utilisable d'une valeur numérique. Par exemple, la chaîne ne doit pas commencer par un blanc :
VAL (" 768") est incorrect
En revanche, VAL ("—768") est correct et donne 768
La fonction contraire de VAL est STR$ qui crée une chaîne de caractères contenant des chiffres :

```
10  D=67
20  PRINT STR$(67)
30  END
RUN
67
```

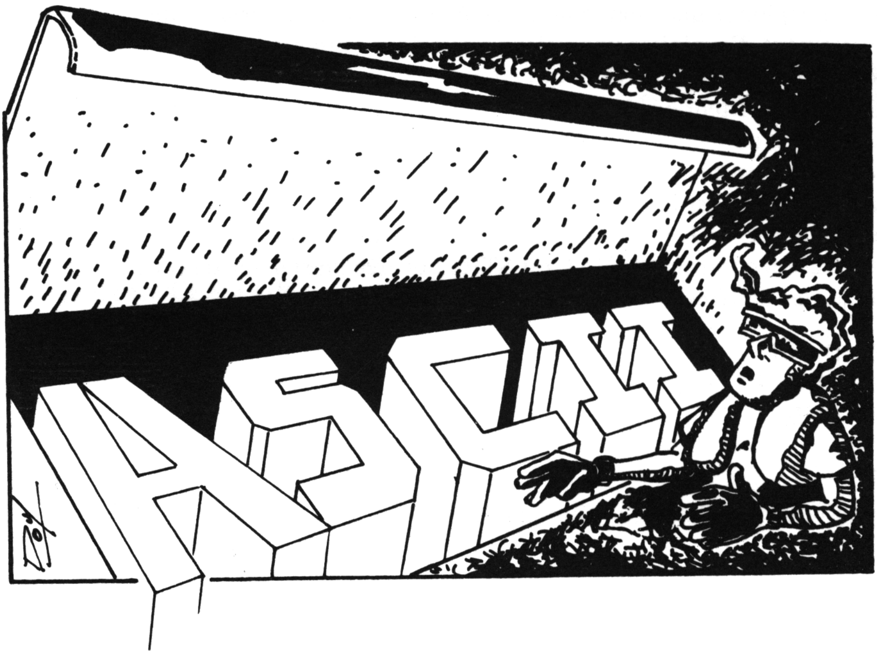
Voici un programme qui utilise cette fonction :

```
5 REM***CHANGEMENT D'ADRESSE***
6 REM
10 CLS
20 INPUT"ADRESSE ACTUELLE: ";AD$
30 Y$=LEFT$(AD$,2)
40 Y=VAL(Y$):PRINT"Y= ";Y
50 Z=Y+(3*2):PRINT "Z= ";Z
60 N$=STR$(Z)
70 A$=N$+RIGHT$(AD$,(LEN(AD$)-2))
80 PRINT "NOUVELLE ADRESSE: ";A$
```

Commentaires : Soit à effectuer un changement du numéro d'une adresse.
30 extraction du numéro de l'adresse dans la chaîne de caractères.
40 transformation numérique du numéro et affichage.
50 changement du numéro. La variable Z prend le nouveau numéro. L'expression 3*2 aurait pu, tout aussi bien, être écrite sous forme de l'entier 6. Affichage de Z.
60 transformation en chaîne de la valeur numérique Z.
70 Recollage de la nouvelle chaîne avec l'ancienne amputée de l'ancien numéro. Utilisation de l'opérateur de concaténation.

Du code ASCII aux caractères

Voici venu le moment de dévoiler le secret des "trucs" de notre esca*mo*teur !



Les caractères (signes, lettres, chiffres) sont présents dans sa mémoire mais sous une forme codée.
Ce code est celui que l'on trouve dans la majeure partie des ordinateurs, le code ASCII (American Standard Code For Information Interchange).
Il est d'une grande simplicité : il associe à chaque caractère un nombre compris entre Ø et 127. (Pour consulter l'ensemble du code ASCII se reporter au tableau de l'annexe numéro trois).
Mais pourquoi ne pas s'adresser directement à l'intéressé ? Une fonction existe qui fera "parler" ATMOS. Elle s'appelle ASC.
?ASC("A")
65
Le numéro de code ASCII de la lettre majuscule A est donc 65
?ASC ("Z")
9Ø
Le code de Z est donc 9Ø
?ASC ("5")
53
Le code du chiffre 5 est 53
?ASC ("!")
33
Le code du point d'exclamation est 33
Pour aller plus vite, voici un petit programme qui vous donnera immédiatement le code du caractère que vos taperez.

```
10 REM CODE ASCII
11 REM
12 FOR I=1 TO 127:REM BOUCLE DU CODE ASCII
15 A$=KEY$
20 INPUT"TAPEZ LE CARACTERE";A$
40 IF A$="" THEN 15
50 PRINTASC(A$)
55 NEXT
60 END
```

Inversement, ATMOS peut vous donner le caractère dont vous lui fournissez le code : adressez-vous à lui avec la fonction CHR$.

? CHR$(65)
A
?CHR$(9Ø)
Z
?CHR$(53)
5
?CHR$(33)
!
?CHR$ (32)
(rien n'apparaît car vous avez tapé le code de la barre d'espacement qui produit un blanc).
En effet, les caractères dont le code est inférieur à 32 n'en sont pas... Ce sont les *caractères "de contrôle"*. Taper ces codes équivaut à une commande ou une instruction.
Essayons :
?CHR$(7)
BEEP (sonnette)
?CHR$(4)
ready
produit le même effet que CTRL D : doublage des caractères.
Nous rencontrerons plus loin d'autres emplois de ces fonctions.
Voici un petit programme qui donne le caractère dont vous entrez le code.

```
10 REM++DECODAGE++
20 CLS
25 FOR I=0 TO 127
30 INPUT"DONNEZ UN CODE ENTRE O ET I27: ";X
40 PRINT "LE CARACTERE EST: ";CHR$(X)
50 NEXT
60 END
```

Opérateurs de relation

Les opérateurs de relation =, <, >, peuvent être utilisés avec les chaînes de caractères grâce au code ASCII. Lorsqu'on compare deux chaînes, les deux premieres lettres sont comparées. Si elles sont identiques, la machine examine les deux suivantes. Une chaîne est dite "inférieure" à une autre dès que l'une des 2 lettres de la comparaison est plus petite (c'est-à-dire plus près du A qui a pour code ASCII 65) que l'autre, ou si, à égalité, l'une des chaînes est plus courte.

Par exemple : "INFORMATIQUE" > "INFORMATICIEN"
En effet > C
(ASCII 81) (ASCII 67)

Ces opérateurs de relation sont utiles dans les tests de réponses ou les tris alphabétiques que nous verrons dans le chapitre "Traitement des données et complément sur les chaînes de caractères".

Encore un truc d'ATMOS : GET



Cette instruction fournit à votre programme le caractère (numérique ou alphabétique) que vous tapez au clavier. Voici un exemple d'utilisation :

```
1Ø  GET X
2Ø  PRINT X
RUN
```

A cet instant, ATMOS attend une donnée numérique d'un caractère. Si vous tapez 2, il affichera ce chiffre. Il en va de même avec une variable chaîne (1) :

```
10  GET X$
20  PRINT X$
RUN
```

(Vous tapez "H", par exemple)
H
ready

Un détail important : tant que vous n'avez pas tapé votre caractère, le programme est bloqué et attend la donnée. Une fois entrée, ce caractère est affiché sans que vous ayez à taper RETURN. Enfin, si vous ne demandez pas l'affichage (PRINT), le caractère n'apparaît pas à l'écran.
HELP ! Cette instruction est un GETpier ! Comment en sortir sans couper le courant ? La solution en fin de chapitre...

Voici un premier programme utilisant les fonctions chaînes :

```
10 REM+++DECOMPTE DES VOYELLES+++
11 REM
15 CLS
16 PAPER 2
20 PRINT "Ce programme compte les voyelles"
30 PRINT"d'une chaine de carateres"
40 PRINT
50 PRINT"Ecrivez un mot ou une phrase"
60 INPUT A$
70 FOR I=1 TO LEN(A$)
80 B$=MID$(A$,I,1)
90 IF B$="A" OR B$="E" OR B$="I"
   OR B$="O" OR B$="U" OR B$="Y" THEN  C=C+1
100 NEXT I
110 PRINT "Nombres de voyelles: ";C
120 END
```

(1) Cette deuxième version est préférable car il acceptera aussi les nombres.

CHECK-LIST

IF... THEN... ELSE 188
FOR... STEP... NEXT 189
PRINT TAB 164

+
=
<
>
< >
LEN 177
LEFT$ 179
RIGHT$ 179
MID$ 178
VAL 182
GET 166
CHR$ 181
ASC 180
STR$ 183

TRAINING

• Écrire un programme combinant les différents éléments soit de mots scientifiques, soit de mots divers, soit de mots de votre invention (concaténation).
• Affichages de chaînes en diagonale, à l'envers, etc. (FOR... NEXT, MID$, RIGHT$, LEFT$).
• Écrire un programme d'orthographe automatique : il s'agira pour ATMOS de mettre le mot proposé par l'utilisateur au pluriel sans se tromper :
  • si le mot se termine par S,X ou Z l'afficher tel quel
  • s'il se termine par OU, lui ajouter un S sauf pour CHOU, CAILLOU, GENOU et ses amis qui prennent un X
  • et s'il se termine par AL, le retirer et le remplacer par "AUX"

Solution à "GETpier" : réécrire le programme avec une instruction de sortie (cf chapitre 1)

La mise au point et la correction des programmes

Les erreurs

Nul n'est parfait et l'erreur est humaine et elle sera toujours humaine, jamais électronique. Vous rencontrerez en gros deux types d'erreurs dans vos programmes :

— celles qui entraînent un message d'erreur. Ce sont les plus évidentes et les plus faciles à trouver. ATMOS est gentil et il vous dit à quelle ligne se trouve l'erreur et de quel type il s'agit. En général, on trouve, avec un peu d'habitude, facilement l'erreur. La liste des messages d'erreurs et de leur signification se trouve en annexe numéro un.

— celles qui donnent des résultats faux ou un déroulement imprévu du programme. C'est plus grave. Votre programme a été mal conçu et il faut le revoir.

C'est plus difficile à détecter aussi car ATMOS ne vous dit rien.

Détection des erreurs

TRON. TROFF

ATMOS ne vous indique pas où se trouve l'erreur mais il peut vous aider en vous disant où il en est dans son programme. A vous de voir si cela correspond à ce qui avait été prévu. C'est le rôle de l'instruction TRON qui signifie TRACE ON, c'est-à-dire marque ma trace. La trace, ce sont les numéros de ligne de programme qui s'affichent sur l'écran au fur et à mesure du déroulement.

Reprenons l'exemple du cercle. Vous étiez fatigué et vous avez écrit :

```
1Ø  INPUT "quel est le rayon du cercle";R
2Ø  P=2×PI×R : S=PI×R↑2
3Ø  PRINT "R=";R, "P=";P, "S=";S
4Ø  IF R< >Ø THEN 1Ø
45  IF R=<Ø THEN PRINT "Attention un rayon est toujours positif":
    GOTO 1Ø
5Ø  END
```

Vous faites tourner le programme avec diverses valeurs de R. Ensuite, vous voulez tester la protection. R = −2

Et ATMOS affiche R, P et S

La protection n'a pas fonctionné.

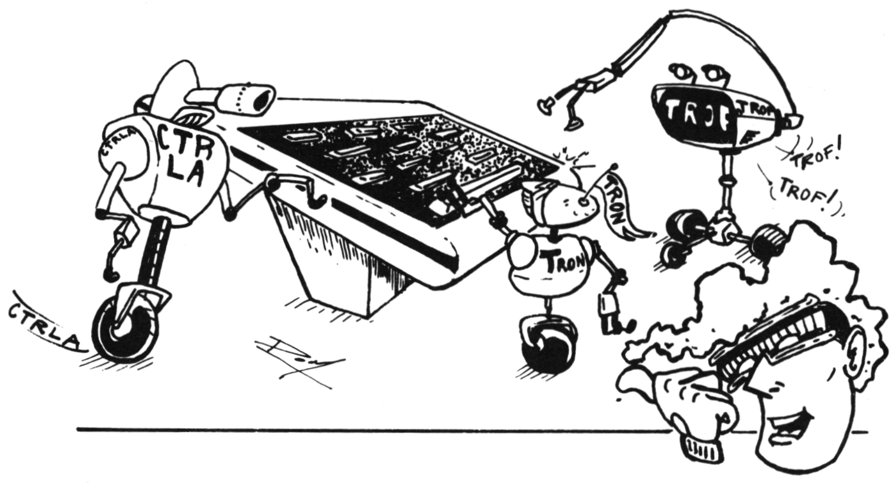
Il est très tard et vous êtes très fatigué !

Alors faites TRON : RUN

Sur l'écran s'affiche le numéro <1Ø> Répondez 1. S'affichent <2Ø> <3Ø> <4Ø> <1Ø> C'est normal.

Répondez −1. S'affichent <2Ø> <3Ø> <4Ø> <1Ø>

Même chose. Ce n'est pas normal. Réfléchissons. Bon sang, mais bien sûr ! Vous avez mis la protection après le test qui renvoie au début. Et comme ce test englobe celui de la protection, on ne passe jamais à la ligne 45. Cette ligne doit être entre 1Ø et 3Ø. On avait simplement tapé 45 au lieu de 25.
Écrivons 25 IF R = <Ø THEN PRINT "Attention un rayon est toujours positif" : GOTO 1Ø
puis 45 RETURN pour effacer la ligne 45
Essayons le programme avec +1 d'abord et −1 ensuite. La protection fonctionne. Arrêtons le programme avec Ø RETURN
Attention un rayon est toujours positif
?
Et ça ne s'arrête pas. Cette machine ne comprend rien à rien !
Restons calme. Voici un exemple de comportement imprévu. Vous pourriez à nouveau "tracer" le programme. Regardons plutôt le listing.
Lorsque R=Ø on ne va pas en 5Ø END mais on revient en 1Ø avec le message. On vient par conséquent de la ligne 25. C'est normal puisque le test de la ligne 25 (IF R= <Ø) est vrai si R=Ø. Il faut bien sûr supprimer = d'où
25 IF R<Ø THEN...
Et le programme est correct.
Bien sûr, utiliser TRON dans ce cas, c'est un peu comme enfoncer un clou avec un marteau pilon.
Un peu de réflexion aurait permis de trouver les erreurs. Mais TRON est souvent très utile pour des programmes plus complexes. Vous pouvez :
— le mettre avant RUN comme vous l'avez fait
— le mettre dans une ligne en début de programme, ici par exemple 5 TRON
— le mettre dans une ligne au milieu du programme pour "tracer" seulement la partie défaillante. Vous supprimerez alors l'effet de TRON en insérant à la fin de cette partie une ligne avec TROFF : trace off.



Des trucs

TRON et TROFF ne sont pas les seuls moyens de détecter les erreurs. Ce sont les ultimes recours.
Très souvent, l'erreur vient d'une variable qui prend une valeur inattendue. Il est souvent fructueux de tester l'état des variables à un moment donné de l'exécution. Vous pouvez le faire.
— Lorsque le programme est en erreur.
En mode immédiat vous faites ? X, ? Y etc. pour les différentes variables et vous comparez aux résultats attendus.
— En introduisant dans le programme, en un point stratégique, une ligne avec STOP qui suspend l'exécution. Vous pouvez alors tester vos variables de la même façon et si tout va bien reprendre l'exécution par CONT (après éventuellement avoir inséré un nouveau STOP un peu plus loin).
— En introduisant des PRINT pour certaines variables dans des endroits bien choisis. (Par exemple une boucle). Cela vous permettra de connaître la vie de vos variables et, sans doute, de trouver la faille.
L'erreur est maintenant trouvée, vous pouvez la corriger. L'ÉDITEUR de l'ATMOS est là pour vous épargner du travail inutile.

La correction du programme avec l'éditeur

L'éditeur ne sert pas uniquement à effectuer des corrections sur une ligne de programme, il permet aussi de créer des lignes qui ressemblent à une ligne déjà écrite ou de fusionner plusieurs lignes.

Correction d'une erreur dans une ligne

Exemple :
Après avoir fait LIST, vous constatez que vous avez écrit :
30 IF A + 16 THEN GOSUB 2000
erreur
La méthode est la suivante :
— On amène le curseur au début de la ligne par les flèches de direction.
— En agissant sur [CTRL] [A], le curseur se déplace à droite. L'effet est très différent de celui de →. Ici, tous les caractères qui sont passés sous le curseur sont mémorisés dans une mémoire tampon. (Attention le caractère sous le curseur n'est pas encore mémorisé. Il faut que le curseur soit à sa droite).
— En appuyant sur [RETURN] la mémoire tampon est transférée dans la mémoire du programme et la ligne est enregistrée.

Ici

Par [↑] ou [←] successifs	[///]	30 IF A + 16 THEN GOSUB 2000
Par [CTRL] [A] successifs		30 IF A[/+/]16 THEN GOSUB 2000
Par [=]		30 IF A = [/1/]6 THEN GOSUB 2000

Rq. En tapant = on considère que le caractère est passé sous le curseur donc est enregistré dans la mémoire tampon.

Par [CTRL] [A] successifs 3Ø IF A = 16 THEN GOSUB 2ØØØ [///]
Par [RETURN] 3Ø IF A = 16 THEN GOSUB 2ØØØ [///]
Refaites LIST et la ligne est corrigée.
Rq. N'oubliez pas, après la correction, de faire [CTRL] [A] jusqu'au bout sinon le reste de la ligne n'est pas mémorisé.

Suppression de caractères

Même ligne : 3Ø IF A = 16 THEN GOSUB 2ØØØ
On veut supprimer GOSUB
Par [↑] [←] [///] 3Ø IF A = 16 THEN GOSUB 2ØØØ
[CTRL] [A] successifs 3Ø IF A = 16 THEN [/G]OSUB 2ØØØ
Ici le G n'est pas mémorisé : le curseur est encore dessus
[→] successifs 3Ø IF A = 16 THEN GOSUB [///]ØØØ
les caractères GOSUB ne sont pas mémorisés
[CTRL] [A] 3Ø IF A = 16 THEN GOSUB 2ØØØ[///]
[RETURN]
Vous avez l'impression que la correction n'a pas été faite.
Faites un LIST
3Ø IF A = 16 THEN 2ØØØ
Rq. DEL est plutôt utilisé en cas d'erreur pour annuler l'effet du [CTRL] [A] sur un caractère. Attention, DEL efface le caractère à gauche du curseur.

Insertion

— D'une ligne : rappelons que les lignes de programme peuvent être tapées dans le désordre.
— En fin de ligne :
Amenez le curseur en début de ligne. Par [CTRL] [A] amenez-le en fin de ligne. La ligne est dans le tampon. Tapez alors ce que vous voulez rajouter. RETURN enregistre le tout.
— Dans une ligne :
Exemple : dans 3Ø IF A = 16 THEN 2ØØØ
Vous voulez A1 au lieu de A
[↑] [←] [///] 3Ø IF A = 16 THEN 2ØØØ
[CTRL] [A] 3Ø IF A[/=]16 THEN 2ØØØ
Si vous tapez le 1 ici, vous perdez le = qui n'a pas encore été mémorisé. Il faut donc revenir en arrière
[←] 3Ø IF[/A] = 16 THEN 2ØØØ
[1] 3Ø IF 1[///]16 THEN 2ØØØ
Le A a disparu mais n'oubliez pas qu'il a été enregistré
[CTRL] [A] jusqu'au bout
[RETURN]
Listez pour vérifier.
Si vous devez insérer plusieurs caractères, plusieurs mots, revenez suffisamment en arrière par ← et faites bien attention de ne pas empiéter sur ce qui n'a pas encore été enregistré.
Si vous voulez que ce soit plus clair, vous pouvez également écrire en dessous ce que vous voulez insérer.

Exemple : 3Ø IF A = 16 THEN 2ØØØ
On veut ajouter après A = 16 "OR B = 24"

[←] [↑] [///] 3Ø IF A = 16 THEN 2ØØØ
[CTRL] [A] 3Ø IF A = 16 [/T/]HEN 2ØØØ
[↓] 3Ø IF A = 16 THEN 2ØØØ
OR B = 24 Puis, ramenez le curseur là où il était
[↑] [←] 3Ø IF A = 16 [/T/]HEN 2ØØØ
OR B = 24
[CTRL] [A] jusqu'au bout et [RETURN]

Fusion de lignes

Si vous avez bien compris, c'est évident. Puisque CTRLA emmagasine tout tant que l'on n'a pas fait RETURN, on peut mettre bout à bout tout ce que l'on veut.
Exemple :
3Ø IF A = 16 THEN 2ØØØ
4Ø IF A = 18 THEN 3ØØØ
Par [CTRL] [A], vous amenez le curseur jusqu'à la fin de 3Ø.
Tapez ":"
Par ↓ et → vous l'amenez après 4Ø sur le I.
Par [CTRL] [A] jusqu'à la fin de 4Ø. [RETURN]
Attention : N'oubliez pas ensuite de supprimer la ligne 4Ø qui est restée intacte par 4Ø [RETURN].

Recopies de lignes

Très souvent dans les programmes, on retrouve des lignes identiques ou très voisines à des numéros différents.

```
3Ø IF A$ = "T" THEN INPUT "T"; T
4Ø IF A$ = "E" THEN INPUT "E"; E
5Ø IF A$ = "D" THEN INPUT "D"; D
```

Après avoir écrit la ligne 3Ø, vous la modifiez par [CTRL] [A] et [E] 3 fois. [RETURN]. (N'oubliez pas de modifier le n° de ligne). Même chose pour 5Ø.
Vous gagnerez un temps précieux.
Vous pouvez de la même façon renuméroter des lignes (n'oubliez pas de supprimer ensuite l'ancienne).

Exécution immédiate d'une commande plusieurs fois

Vous savez que vous pouvez utiliser l'ordinateur en mode direct (on dit aussi en mode machine de bureau). C'est très utile pour calculer certaines fonctions ou pour tester certaines lignes de programme.

L'éditeur vous permet d'exécuter plusieurs fois une ligne. Exemple : vous voulez calculer une fonction cosx + $\sin^3$ x plusieurs fois pour des valeurs différentes.

? cos(1.22) + (sin (1.22))↑3
1.17184461

Si vous voulez la même fonction pour une autre valeur, repassez sur la ligne avec [CTRL] [A] en modifiant l'intérieur des parenthèses.

Vous pouvez aussi tester de cette façon une ligne de programme en mode direct : il suffit de repasser sur la ligne avec [CTRL] [A] en évitant le n° de ligne.

Exemple :

IF X=Ø THEN PRINT "NUL"; ELSE PRINT "NON NUL"'
RUN
NUL
? SYNTAX ERROR IN 5Ø

Essayez par des essais directs de trouver d'où vient l'erreur de syntaxe. (Solution dans la fiche de IF THEN ELSE,)

Et un jeu pour animer vos soirées. Il utilise les acquis des chapitre précédent.

```
10 REM+++ALPHABET+++
15 CLS
16 PAPER2
20 PRINT"JE VAIS PENSER";
21 PRINT" A UNE LETTRE DE L'ALPHABET"
30 WAIT 200
40 PRINT"ESSAYE DE LA DEVINER"
45 WAIT 150
46 PRINT
50 PRINT"Je te donnerai des conseils,";
55 PRINT"bien sur!"
60 WAIT 200
70 PRINT "Par exemple:'TROP BAS',si tu es";
71 PRINT" trop pres de la lettre A"
80 PRINT
85 PRINT"Ou bien 'TROP HAUT'si tu es";
86 PRINT" trop pres de la lettre Z".
90 WAIT 400
100 INPUT"AS-TU COMPRIS(O/N): ";R$
110 IF R$="O" THEN 200 ELSE 120
120 IF R$="N" THEN 10
130 GOTO 100
200 L=65 +INT(RND(1)*26)
210 G=0
230 PRINT:PRINT"COMMENCE A CHERCHER"
240 PRINT:PRINT"QUE PROPOSES-TU";
250 G=G+1
255 INPUT A$:A=ASC(A$):PRINT
260 IF A=L THEN 350
```

```
270 IF A>L THEN 320
300 PRINT "TROP BAS,";
301 PRINT"ESSAYE UNE LETTRE PLUS HAUTE"
305 GOTO 240
320 PRINT"TROP HAUT,ESSAYE UNE LETTRE";
325 PRINT" PLUS BASSE"
330 GOTO 240
340 PING
350 PING:PRINT:PRINT"TU L'AS TROUVEE"
351 PRINT"EN ";G;" ESSAI(S)!"
352 WAIT 300
355 IF G<=5 THEN 400
360 PRINT"MAIS CELA NE DEVRAIT PAS"
361 PRINT"PRENDRE PLUS DE CINQ ESSAIS!"
365 WAIT 200
370 GOTO500
400 PRINT"BIEN JOUE!"
450 WAIT 200
500 CLS:PRINT"ENCORE UNE PARTIE(O/N)"
501 INPUT R$
510 IF R$="O"THEN 200 ELSE 600
600 CLS:PRINT"A UNE AUTRE FOIS..."
610 END
```

CHECK-LIST

TRON, TROFF
STOP, CONT
[CTRL] [A]
[DEL]
[RETURN]
↑→↓←
? (Mode direct ou bureau)
WAIT

TRAINING

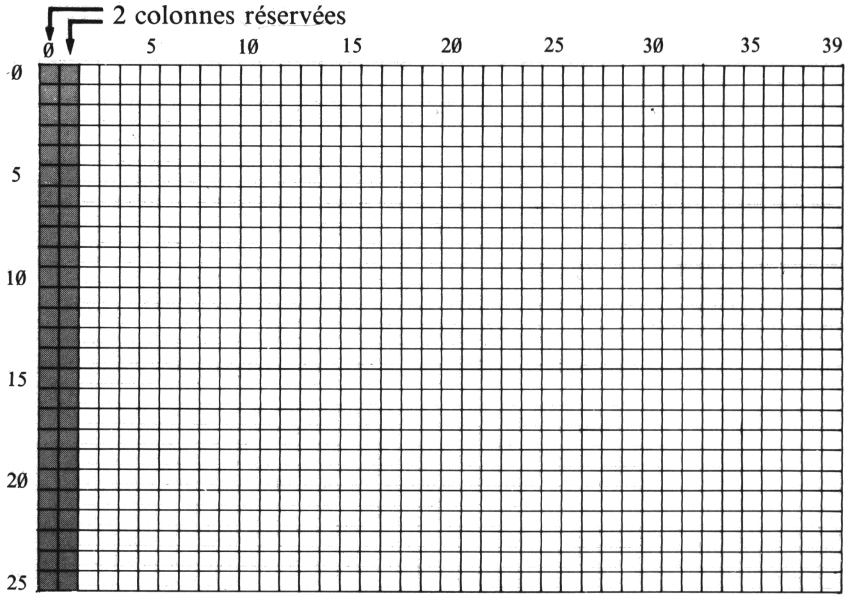
• Utilisez systématiquement l'éditeur dans les programmes qui suivent.

Couleurs et dessins

Nous avons jusqu'ici utilisé uniquement des caractères. On dit que l'on travaille en mode texte (TEXT). ATMOS possède deux autres modes qui permettent des dessins et des couleurs. Ce sont les modes basse résolution et haute résolution (LORES et HIRES).
Nous allons voir successivement les fonctionnements de chacun de ces modes.

Mode TEXT

1. *Géographie de l'écran*
L'écran comprend 4Ø colonnes de 27 caractères. Les 2 colonnes à gauche, ne peuvent être utilisées pour du texte.
La ligne du haut est réservée à des messages de l'ordinateur : CAPS (majuscules) searching, loading (pendant l'utilisation de la cassette). Il reste 38 colonnes de 26 caractères utilisables en écriture normale.



Mode basse résolution (TEXT-LORES)

2. *Couleurs d'une ligne*
Vous pouvez choisir la couleur du fond par PAPER et des caractères par INK.
Mais vous pouvez aussi choisir la couleur d'une ligne (fond et caractères).
Remplissons l'écran par ce petit programme :

```
 5  REM A
1Ø  PAPER Ø : INK 2 : CLS
2Ø  FOR I=1 TO 25
3Ø  PRINT "AAA..." (38 caractères A)
4Ø  NEXT I
```

L'écran se remplit de A.

En mode direct

Amenez le curseur devant une ligne. Tapez [ESC] T (attention, touche [ESC] et ensuite T). Le fond de la ligne se colore en bleu.
ESC A : les caractères deviennent rouge.
Vous avez pu constater :
— que les 2 premières colonnes servent à définir les couleurs du fond et des caractères de la ligne.
— la commande se fait par [ESC] [CAR]: CAR est le caractère de contrôle :

ESC	@	caractères en noir
	A	rouge
	B	vert
	C	jaune
	D	bleu
	E	magenta
	F	cyan
	G	blanc

ESC	P	fond noir
	Q	rouge
	R	vert
	S	jaune
	T	bleu
	U	magenta
	V	cyan
	W	blanc

Vous pouvez remarquer que les caractères de H à O ne commandent pas les couleurs. Ils agissent sur la hauteur ou la forme de l'affichage. Nous les verrons plus tard.

Par programmation

Vous pouvez bien sûr le faire dans un programme. Rappelez-vous que tous les caractères ont un code. ESC a le code ASCII 27.
Pour faire l'équivalent de ESC A dans un programme, écrivez PRINT CHR$ (27) "A"
Rajoutez au programme précédent :

```
45  PRINT CHR$ (3Ø)
5Ø  FOR I=1 TO 12
6Ø  PRINT CHR$ (27) "U" CHR$ (27) "A": PRINT
7Ø  NEXT I
8Ø  GOTO 8Ø
```

Une ligne sur deux est colorée avec un fond magenta et une encre rouge.
En effet, la ligne 6Ø équivaut à :
PRINT CHR$ (27) "U" CHR$ (27) "A": PRINT
ESC U
fond magenta

ESC A
fond rouge

saut de la ligne
ligne suivante
pas modifiée

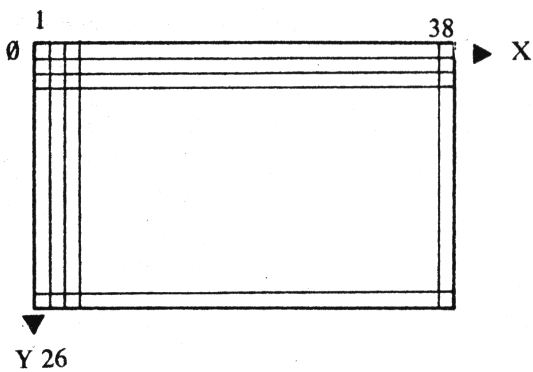
la ligne 45 est indispensable : elle ramène le curseur en haut.
Encore plus joli : Remplacez les lignes 5Ø et 6Ø par

```
5Ø  FOR I=1 TO 25
6Ø  PRINT CHR$ (27) CHR$ (8Ø + INT (8*RND(2))
    CHR$ (27) CHR$ (64+INT (8*RND(2))
```

Et vous verrez toutes les lignes se colorer (encre et papier de façon aléatoire).

```
5 REM A
1Ø PAPER Ø:INK 2:CLS
2Ø FOR I=1 TO 25
3Ø PRINT"AAAAAAAAAAAAAAAAAAAAAAAAAAAA";
35 PRINT"AAAAAAAAAA"
4Ø NEXT I
45 PRINTCHR$(3Ø)
5Ø FOR I=1 TO 25
6Ø PRINT CHR$(27)CHR$(8Ø+INT(8*RND(2)));
65 PRINT CHR$(27)CHR$(64+INT(8*RND(2)))
7Ø NEXT I
8Ø GOTO 8Ø
```

Impression



L'instruction PRINT est très utile mais elle ne permet que des impressions les unes à la suite des autres de la gauche vers la droite et du haut vers le bas.
Il est souvent nécessaire d'imprimer là où on veut et si possible de la couleur que l'on veut. C'est le rôle de PLOT.
Reprenons la géographie de l'écran.
Pour écrire là où on veut, il faut se repérer.
Chaque case est repérée par ses coordonnées X et Y. Attention l'axe des Y est inversé.

Par exemple, la case 1,Ø est en haut à gauche, la case 38,26 en bas à droite.
PLOT X,Y "..." permet de placer un caractère à la case X,Y, ou une suite de caractères commençant à la case X,Y.
Exemple : PLOT 1,Ø,"A"
Vous voyez un A apparaître en haut à gauche
PLOT 38,26,"B"
un B en bas à droite
PLOT 14,13, "ORDINATEUR"
ORDINATEUR apparaît au milieu de l'écran
Le texte peut dépasser la fin de ligne sans erreur mais il empiète alors sur les 2 colonnes de gauche et vous aurez des problèmes de couleur.

Choix des couleurs : notion d'attribut

Nous avons vu dans les paragraphes précédents le rôle de [ESC] [lettre] qui permet de modifier la couleur d'une ligne.
Par exemple ESC T imprimé dans une ligne, soit en mode direct soit par PRINT CHR$ (27) "T" colore le reste de la ligne en bleu.
On peut faire la même chose avec PLOT. Les deux premiers paramètres sont inchangés. Le troisième est maintenant un nombre compris entre Ø et 31. C'est ce que l'on appelle l'attribut.
Ainsi PLOT, 1Ø, 1Ø, 19 aura le même effet que si vous amenez le curseur à la case 1Ø,1Ø et que vous tapez ESC S.
Il y a donc une correspondance directe entre les ESC et les attributs. La voici pour ceux qui ont déjà été vus.

	ATTRIBUT	
ESC@	Ø	encre noire
A	1	rouge
B	2	vert
C	3	jaune
D	4	bleu
E	5	magenta
F	6	cyan
G	7	blanc

ESC P	16	papier noir
Q	17	rouge
R	18	vert
S	19	jaune
T	20	bleu
U	21	magenta
V	22	cyan
W	23	blanc

Essayons :
PLOT 1Ø,1Ø,1 : PLOT 1Ø,1Ø, "H"
Le H s'affiche en noir. Ça ne va pas. Erreur : nous avons oublié que l'attribut prend la place d'un caractère. Si on écrit H au même endroit, il disparaît.
Il faut donc toujours mettre le ou les attributs à gauche du caractère ou de la chaîne que l'on veut modifier.
Ex. :
PLOT 9,1Ø,1: PLOT 1Ø,1Ø, "H"
Et vous obtenez un beau H en rouge.
PLOT 13,14,5: PLOT 14,14, "CHAINE"
et CHAINE s'affiche en magenta.
Vous pouvez mettre plusieurs attributs à condition de les mettre dans des cases différentes, avant la chaîne.
PLOT 12,14,18 : PLOT 13,14,5: PLOT 14,14, "CHAINE"
Le premier PLOT donne un fond vert, le 2e des caractères magenta, le 3e affiche le message.
Remarquez que le fond reste vert jusqu'à la fin de la ligne. Si vous voulez supprimer le fond vert après le message, vous rajouterez à la suite PLOT 24,14,23 si le fond est blanc.

Autres attributs

Les attributs permettent aussi le clignotement, l'affichage en double hauteur, et l'affichage de caractères graphiques.
Le clignotement (12) ou (ESCL) ne pose guère de problème, par exemple :

1Ø PRINT "UN ATTRIBUT" CHR$(27) "L PREND LA PLACE "CHR$(27) "H D'UN CARACTÈRE"

RUN et PREND LA PLACE CLIGNOTE au milieu de la phrase.

CHR$(27) L fait clignoter les caractères jusqu'à CHR$(27) H qui donne un affichage normal.
De la même façon, PLOT 1Ø,1Ø,12: PLOT 11,1Ø, "ATMOS" produira un ATMOS clignotant.
L'affichage en double hauteur (1Ø, ESCJ) est un peu plus délicat.
Sur les 2 lignes où apparaîtra le message, il faut indiquer l'attribut et écrire le message.

Exemple :

```
10  CLS
20  PLOT 12,7,10: PLOT 13,7, "DOUBLE"
30  PLOT 12,8,10: PLOT 13,8, "DOUBLE"
```

et vous aurez DOUBLE en double hauteur.
Attention : il faut que la 2e ligne soit une ligne paire. Essayez par exemple en remplaçant 7 par 8 et 9 dans le 2e paramètre de PLOT.
Si vous écrivez par un PRINT, vous pouvez éviter d'écrire en double par ?CHR$(4) qui le fera à votre place. Vous utiliserez alors CHR$(27) "J"
Exemple :

```
40  PRINT CHR$(4)
50  PRINT "  "CHR$(27)" J DOUBLE HAUTEUR"
60  PRINT CHR$(4)
```

Le 2e PRINT CHR$(4) ramène à l'affichage normal.
En 50, les deux blancs affichés au début évitent d'écrire ESCJ dans les colonnes fond et encre. Le résultat serait alors différent.
Remarquons qu'il existe d'autres attributs permettant de combiner les précédents (voir Annexe 6) ou bien d'afficher des caractères graphiques. Nous en parlerons dès le paragraphe "basse résolution".
Pour conclure, ESC et les attributs jouent exactement le même rôle. ESC sera utilisé en mode direct ou en combinaison avec des ordres PRINT sous la forme CHR$(27), la lettre sera incluse dans le message.
PLOT avec un attribut sera utilisé avec d'autres PLOT. Faites attention de les mettre avant le message.
L'instruction PLOT sera très utilisée dans le mode LORES dans un but un peu différent.
C'est ce que nous allons voir maintenant.

Modes LORES

LORES signifie basse résolution : cela signifie que l'on va dessiner mais avec des gros points et des gros traits.
Le mode LORES comprend 2 modes : LORES 0 et LORES 1. LORES 0 ressemble beaucoup à TEXT, LORES 1 n'a plus de caractères alphanumériques.

Mode LORES 0

On y vient en tapant LORES 0 (qui est aussi une instruction).
L'écran devient noir.
La principale différence avec le mode TEXT vient de l'effet des attributs.
Essayons PLOT 10,10,20
Nous voyons apparaître un carré jaune de coordonnées 10,10. Rappelez-vous qu'en mode TEXT, cela colorait toute la ligne.
Si vous faites maintenant PLOT 11,10,20 apparaît à droite un carré bleu.
Vous découvrez immédiatement l'intérêt de ce mode : on peut tracer où on veut de petits carrés de la couleur que l'on veut.

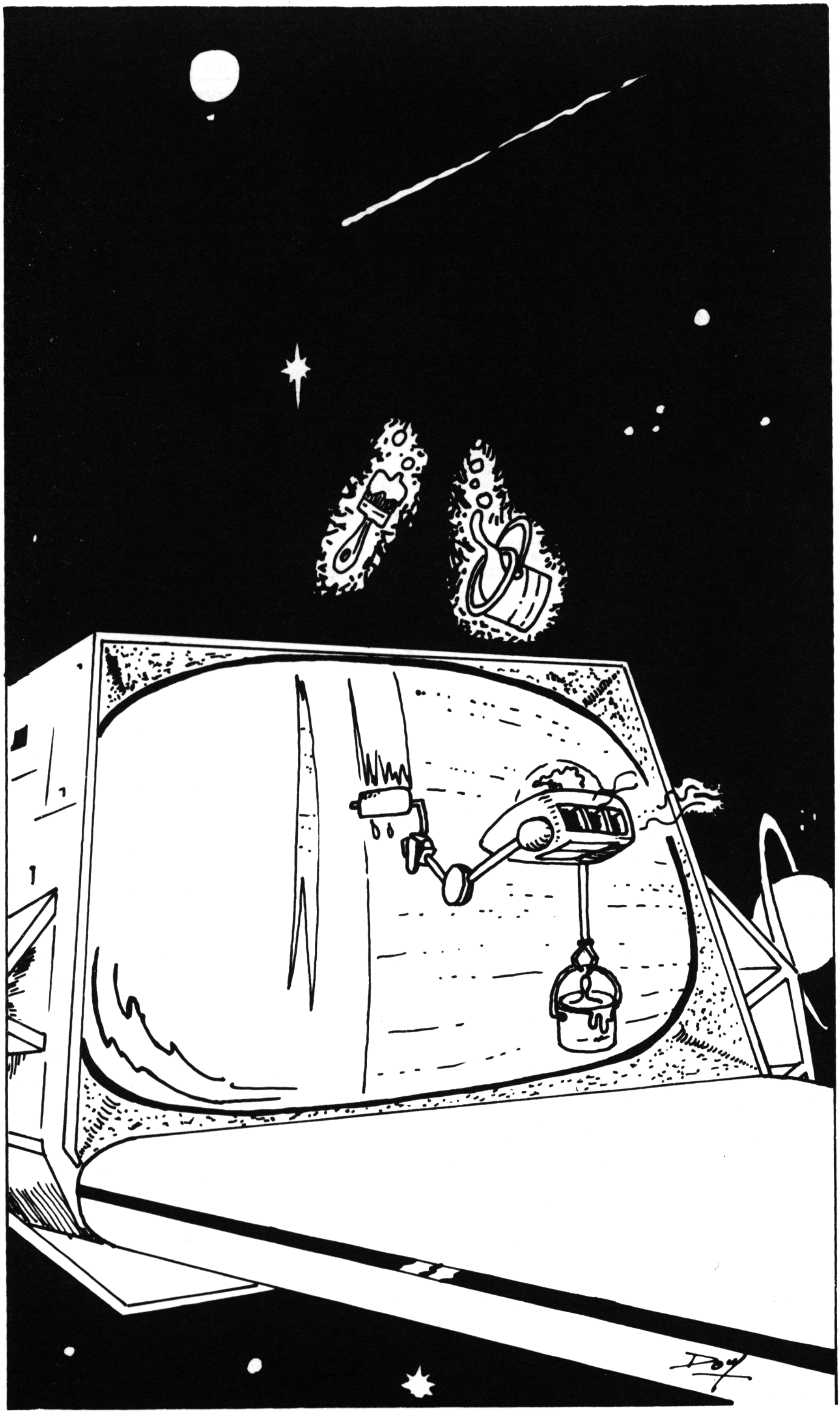
Par exemple :

```
11 REM
20 LORES 0
30 FOR I=1 TO 38
40 FOR J=0 TO 26
50 PLOT I,J,16+I+J-INT((I+J)/8)*8
60 NEXT J:NEXT I
70 GOTO 70
```

Pas mal ! N'est-ce pas ?
Vous trouvez cela monotone. Alors remplacez 5Ø par :
5Ø PLOT I,J,16 + 8 * RND(2)
Tous les goûts sont dans la nature.
Et celui-ci :

```
5 REM+++RECTANGLES+++
10 CLS
20 FOR P=0 TO 12 STEP2
30 FOR X=P+1 TO 38-P
40 PLOT X,P,17+P/2
50 PLOT X,26-P,17+P/2
60 NEXT X
70 FOR Y=P TO 26-P
80 PLOT P+I,Y,17+P/2
90 PLOT 38-P,Y,17+P/2
100 NEXT Y
110 NEXT P
120 END
```

Et si vraiment vous voulez faire de jolis dessins géométriques voici maintenant un mini-éditeur graphique.
Les commandes sont les suivantes :
Touches [Ø] à [7] choix des couleurs (code habituel)
Touches [←] [↓] [↑] [→] déplacement du curseur
Touche [DEL] mode effacement
Touche [E] effacement de tout l'écran
Touche [barre espace] barre espace mode déplacement. Le curseur se déplace sans modifier le dessin.
Touche [S] sauvegarde de l'écran sur cassette.
Touche [L] chargement d'un dessin à partir d'une cassette.
Pour la compréhension complète du programme, attendez d'avoir lu l'ensemble du GUIDE.



```
5 REM+++EDIDESS+++
10 CLS:LORES0
20 X=19:Y=13:P=7
30 GOSUB 2000
50 C=SCRN(X,Y)
60 PLOT X,Y,16:WAIT 10
70 PLOT X,Y,23:WAIT 10
80 PLOT X,Y,C
100 A$=KEY$:IF A$=" THEN 50
110 A=ASC(A$)
130 IF A8 THEN 800
140 IF A=9 THEN 900
150 IF A=10 THEN 1000
160 IF A=11 THEN 1100
170 IF A=32 THEN 1200
180 IF A=127 THEN 1300
190 IFA>47ANDA<56THENP=A-48:B=0:GOSUB2000
200 IFA$="S"THEN GOSUB 3000
210 IFA$="L"THEN GOSUB 3100
300 GOTO 50
800 X=X-1
810 IF X<1 THEN X=1:GOTO 50
820 GOSUB2000
830 GOTO 50
900 X=X+1
910 IF X>38 THEN X=38 GOTO 50
920 GOSUB 2000
930 GOTO 50
1000 Y=Y+1
1010 IF Y>26 THEN Y=26:GOTO 50
1020 GOSUB 2000
1030 GOTO 50
1100 Y=Y-1
1110 IF Y<0 THEN Y=0:GOTO 50
1120 GOSUB 2000
1130 GOTO 50
1200 B=1:GOTO 50
1300 B=0: P=0:GOSUB 2000 :GOTO 50
2000 IF B=1 THEN RETURN
2010 PLOT X,Y,16+P
2020 RETURN
3000 INPUT"NOM DU DESSIN";N$
3010 CSAVE N$,A#BB80,E#BFDF,S
3020 RETURN
3100 INPUT"NOM DU DESSIN";N$
3110 CLOAD N$,A#BB80,E#BFDF,S
3120 RETURN
```

Mode LORES 1

Tapez LORES 1. [RETURN]
Utilisez alors le clavier normalement. Plus de caractères. Ils sont remplacés par des motifs géométriques. C'est le deuxième jeu de caractères de l'ATMOS.
Voulez-vous tous les voir ?
Facile. Les commandes ESC ? CHR$(27) n'ont plus de secret pour vous et vous n'aurez aucun mal à comprendre le programme suivant qui vous donne à la suite : le code ASCII, le caractère normal et le caractère alterné.

```
10  CLS: PAPER Ø : INK 2
20  FOR I=32 TO 127
30  PRINT CHR$(27) "H"I;CHR(I); CHR(27) "I"; CHR(I); "     ";
40  NEXT I
```

Vous aurez au préalable pu vérifier que ESC I fait passer le clavier en caractères graphiques et ESC H en caractères normaux. Le programme est alors évident.
A vous d'imaginer des dessins qui mettent en œuvre ces caractères graphiques. Nous verrons plus loins que si ces caractères ne vous plaisent pas, vous pouvez les transformer à volonté pour les adapter à vos goûts.

CHECK-LIST

TEXT LORES Ø LORES 1 200 201
[ESC] caractère...
Notion d'attribut ou de caractère préfixé
PLOT 165

TRAINING

- Programme qui dessine une maison.
- Programme qui dessine une maison sur de l'herbe (Mode LORES 1 pour dessiner l'herbe).
- Programme qui dessine des triangles.

Bruits, son, musique

L'ATMOS dispose d'un synthétiseur de son qui permet des effets surprenants.



Sons obtenus par le clavier

— Toutes les touches pressées produisent un bip (sauf [SHIFT]).
— Le son est plus grave pour [RETURN] [ESC] [CRTL] et les flèches de positionnement du curseur.
Les bip du clavier sont supprimés si vous agissez sur [CTRL] [F] (si ça agaçe vos voisins). Vous les faites réapparaître de la même façon.
— [CTRL] [G] produit un DING très cristallin.

Fonctions préprogrammées

1. Tapez PING : vous retrouvez le son du CTRL G.
De la même façon, ZAP produit le son du pistolaser le plus ordinaire alors que SHOOT et EXPLODE nous ramènent au temps des cowboys et des indiens.
2. Ces quatre sons sont des instructions qui seront incluses dans les programmes.

Essayons :
10 PING
20 SHOOT
30 EXPLODE
40 ZAP
RUN. Que se passe-t-il, on n'entend que le ZAP ?
En réalité, chaque instruction arrête la précédente. Et comme le déroulement est très rapide, on n'entend que la dernière. Il suffit d'ajouter une temporisation qui permet au son de se terminer.
Ajoutons en 15, 25 et 35 WAIT : 100 et on assiste à la bataille.
Vous verrez par la suite que WAIT est très utilisée pour les programmes musicaux. (Rappelons que le paramètre après WAIT indique le nombre de centièmes de secondes de la temporisation).

Les instructions pour la musique et le son

Le générateur de sons comporte 6 canaux.
— Les canaux 1, 2 et 3 sont réservés à la musique et aux sons purs.
— Les canaux 4, 5 et 6 sont mélangés avec un bruit.
Les instructions sont au nombre de 3.
SOUND et MUSIC définissent un son ou une note sur un canal.
PLAY a deux rôles :
— Elle ouvre ou ferme les canaux avec possibilité d'en ouvrir plusieurs à la fois. (En utilisant les 2 premiers paramètres).
— Elle permet de moduler la sortie (enveloppe programmable par les 2 derniers paramètres).

Création d'un son

Essayez 10 SOUND 2, 500, 8. Ce qui signifie : Je crée un son sur le canal 2 de période 500 et de niveau 8 (remplacez 8 par 4 après 22 heures ou par 14 si vous êtes dur d'oreille).
RUN. Il ne se passe rien. Pourquoi ? Parce que le canal n'a pas été ouvert (c'est comme un robinet).
Ajoutez :
20 PLAY 2, 0, 0,0.
2 signifie que l'on ouvre le canal 2. Les 3 autres paramètres seront vus ultérieurement.
RUN. Et vous voilà dans la station spatiale SPACE ATMOS 237.
Pour en sortir, appuyez sur une touche quelconque ce qui ferme le canal.
Vous auriez pu rajouter :
30 WAIT 1000
40 PLAY 0, 0, 0, 0.
Après 10 secondes, le son est interrompu. Attention, comme vous l'avez constaté, l'instruction PLAY n'arrête pas le déroulement du programme. Le son continue tant que le programme ne rencontre pas un nouveau PLAY qui l'arrête ou le modifie.

Période

Pour tester l'influence du 2e paramètre de SOUND (période), essayez le petit programme suivant :

```
10  INPUT T
20  SOUND 2, T, 8
30  PLAY 2, 0, 0, 0
40  GOTO 10
```

Remarquons que le INPUT demande un nombre. Le fait d'appuyer sur une touche pour introduire ce nombre arrête le son.
Plus T est grand et plus le son est grave. Il devient audible à partir de 4 ou 5. Après 4095, vous retrouvez les mêmes sons.

Mélange de sons

Le premier paramètre permet de choisir les canaux 1, 2 et 3 et de les mélanger. Essayez à partir du programme précédent :

```
 5  REM Mélange
10  INPUT T
20  SOUND 2, T, 8
21  SOUND 1, 2*T, 8
30  PLAY 2, 0, 0, 0
35  WAIT 300
40  PLAY 1, 0, 0, 0
45  WAIT 300
50  PLAY 3, 0, 0, 0
55  WAIT 300 : GOTO 10
```

Vous entendez successivemet un son (canal 2), un son de période double (canal 1) et les deux ensembles (canal 1 + 2). Attention au codage du 1er paramètre. 3 signifie canaux 2 et 1 ouverts et non pas canal 3 ouvert (voir les fiches de référence).

Les sons et les bruits

Dans le programme précédent remplacez le 1er paramètre de SOUND par 4 et remplacez la ligne 30 par PLAY 0, 1, 0, 0
Exécutez, vous entendez pour T = 2 un bruit rappelant vaguement celui d'un torrent (quelle imagination !).
Les canaux 4, 5 et 6 comportent tous ce bruit qui dépend de la période T imposée par SOUND.
Le deuxième paramètre de PLAY permet d'ouvrir ou de mélanger les trois canaux.

Musique

Si vous ne supportez pas le bruit, l'instruction MUSIC est pour vous. Elle définit une note sur un canal.

Exemple :

```
10  INPUT N
20  MUSIC 1, 2, N, 8
30  PLAY 1, 0, 0, 0
40  GOTO 10
```

vous permettra de jouer les notes de la 2[e] octave. Vous pouvez ainsi transformer votre clavier en orgue électronique.

```
1 REM+++ORGUE+++
2 PAPER4
5 CLS :O$="3"
10 PRINTCHR$(6)
20 A$=KEY$:IF A$="" THEN 20
30 IF A$="O" THEN GET O$
35 IF ASC(O$)<49 OR ASC(O$)>55 THEN O$="3"
40 IF A$="0" THEN A$="10"
50 IFA$="-" THEN A$="11"
60 IF A$="=" THEN A$="12"
65 IF A$=CHR$(13) THEN 66 ELSE 70
66 PLAY 0,0,0,0:PRINT CHR$(6):END
70 IF ASC(A$)>57 OR ASC(A$)<49 THEN 20
100 MUSIC 2,VAL(O$),VAL(A$),10
110 PLAY 2,0,0,0
120 PRINTVAL(A$);:GOTO 20
```

Les touches du haut servent à jouer (de [1] à [=])
Pour changer d'octave, tapez 0 puis le numéro de l'octave.
Pour arrêter, appuyez sur [RETURN].
Quelques indications sur le programme lui-même :
En 10, on supprime le BIP du clavier (voir annexe sur les caractères de contrôle).
De 20 à 70, il s'agit de la boucle de saisie du clavier.
30 entraîne, si on appuie sur 0 un GET qui vous permet de modifier l'octave.
35 est une protection en cas de chiffre interdit pour l'octave.
40-50—70 permettent de saisir les notes.
65 correspond à la touche RETURN qui entraîne un PLAY0,0,0,0 pour arrêter la musique et un ? CHR$ (6) pour restaurer le BIP du clavier avant arrêt.
Le reste ne présente pas de difficulté.
Vous trouvez peut-être le résultat quelque peu décevant : Nous ferons mieux quand vous serez plus savant.

Enveloppe de sortie

Jusqu'ici, le volume des sons et des notes étaient fixes. (3e paramètre de SOUND ou 4e de MUSIC). D'où une certaine monotonie.
Les deux derniers paramètres de PLAY vous permettent d'agir sur l'enveloppe de sortie à condition que le paramètre volume de SOUND ou MUSIC ait été mis à Ø.
Exemple :

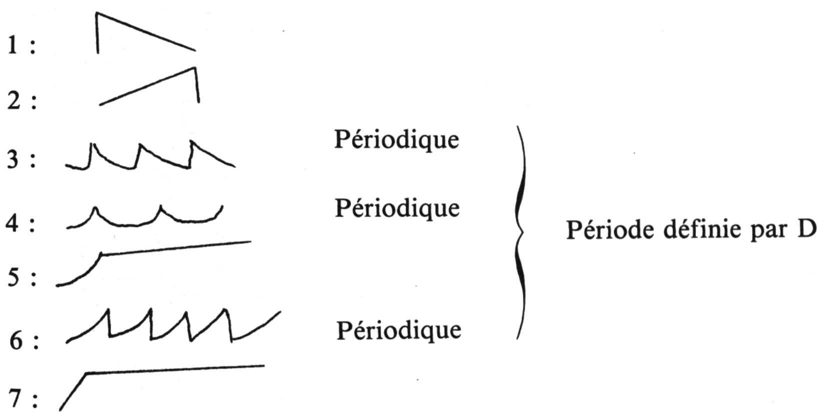
```
3Ø  MUSIC 2, 2, 4, Ø
4Ø  PLAY 2, Ø, 2, 32ØØØ
```

Le son créé par MUSIC est modulé par PLAY en fonction des paramètres
3e E : ENVELOPPE
4e D : DURÉE

Dans le programme précédent :
E = 2. D'où un volume croissant pendant une durée définie par D. Ici D = 32ØØØ soit environ 15 s.

On dispose de 8 enveloppes différentes dont 3 sont périodiques.

Ø : Enveloppe constante. Niveau défini par Sound ou Music



Cette modulation peut intervenir pour SOUND ou pour MUSIC.

Essayons :

```
2Ø  SOUND 1, 8ØØ, Ø
3Ø  PLAY 1, Ø, 4, 3ØØ
```

Nous voici de retour dans l'espace.
Il est difficile de prévoir vraiment le son que l'on va obtenir. Le plus simple est de faire des essais systématiques. Voici un petit programme qui vous aidera.

```
5 REM+++SONS+++
10 CLS
20 A$=KEY$
30 IF A$="T"THEN INPUT "T";T
40 IF A$="E"THEN INPUT "E";E
50 IF A$="D"THEN INPUT "D";D
52 IF A$="1" THEN C=NOT C
53 IF A$="4" THEN B=NOT B
55 IF A$="" THEN 20
60 SOUND 1,T,0
70 PLAY ABS(C),ABS(B),E,D
75 PRINT"T:";T;"     E:";E;"     D:";D;
77 IF C=-1 THEN PRINT"     1";
78 IF C=0 THEN PRINT"      ";
79 IF B=-1 THEN PRINT"     4"
80 IF B=0 THEN PRINT"      "
90 GOTO 20
```

Ce programme un peu plus long amène quelques commentaires.
1. Ne vous découragez pas. Beaucoup de lignes se ressemblent et vous gagnerez du temps en utilisant l'éditeur pour modifier une ligne existante.

2. Le programme est formé de 3 modules :
— *Le module son, 60 à 75.* Vous trouverez 5 paramètres variables T, C, B, E, et D. Ces 5 paramètres pouvant être modifiés grâce au module scrutation.
— *Le module scrutation et entrées, 20 à 55.* La fonction KEY$ permet de saisir le clavier au vol. Contrairement à GET, elle n'arrête pas le déroulement du programme.
— *Le module affichage, 77 à 80.* Pour savoir les valeurs des paramètres et les canaux utilisés.
Essayez-le c'est facile.
Si vous appuyez sur 1, vous ouvrez ou fermez le canal 1 où est créé le son.
Si vous appuyez sur 4 vous ouvrez ou fermez le canal 4 qui vous apporte du bruit.
Si vous appuyez sur T, on vous demande une nouvelle valeur T. (Période) N'oubliez pas de valider.
Même chose pour E (de 0 à 7) Enveloppe,
et pour D (de 0 à 32767) Durée.
Si le son n'est pas répété (E = 1, 2, 5 ou 7) appuyez sur [RETURN].
Sur l'écran s'affichent :
T : 40 E : 2 D : 444 1 4
1 et 4 signifient que les canaux 1 et 4 sont ouverts. Si vous appuyez sur 4, le 4 disparaît de l'écran ainsi que le bruit.
A vous de jouer.
Ainsi avec T : 900 E : 6 D :100 1 4
Diminuer D : 50 20 10 5
Avez-vous vérifié votre parachute ?
Vous pouvez aisément adapter ce programme pour un programme musical qui vous permettra de faire la synthèse de différents instruments.

Il vous suffira de changer le SOUND en MUSIC et de modifier l'entrée de T pour pouvoir entrer des notes. (Vous pouvez vous inspirer du programme orgue).

Voici un exemple améliorable :

```
5 REM  *MUSIC**
10 CLS
15 N=1
20 A$=KEY$
30 IF A$="N" THEN GET N$:N=VAL(N$)
40 IF A$="C" THEN GET C$:C=VAL(C$)
50 IF A$="E" THEN GET E$:E=VAL(E$)
60 IF A$="D" THEN INPUT "D",D
70 IF A$="" THEN 20
100 MUSIC 1,2,N,0
110 MUSIC 2,3,N,0
120 MUSIC 3,4,N,0
130 PLAY C,0,E,D
140 PRINT"CANAL";C;"    E";E;"    D";D
150 GOTO20
```

Si la musique vous passionne, nous vous conseillons le chapitre "NIVEAU, FRÉQUENCE, ENVELOPPE" mais il vous faudra, au préalable, faire des progrès en programmation.

CHECK-LIST

PING 209
ZAP 208
SHOOT 209
EXPLODE 209
WAIT 187
SOUND 209
MUSIC 210
PLAY 211

TRAINING

- Programme qui joue un air connu.
- Le même mais en variant les sonorités.
- Transformez votre clavier en instrument de musique avec enveloppe et durée réglables.

Croisière HYPER-BASIC

Structure de donnée - fonctions mathématiques et opérateurs logiques

Les tableaux

Exemple

Prenons l'exemple d'un météorologue qui relève les températures tous les matins à huit heures : le premier jour il relève une température T1, le deuxième une température T2 et le troisième T3 etc.
Il pourra bien sûr appeler T1, T2, T3... les variables qui contiendront les différentes températures. Ces variables contiennent un indice mais il ne s'agit pas d'un véritable indice ; un exemple, supposons que une variable N contienne le numéro du jour, donnez un programme qui donne la température ce jour-là. Ce n'est pas facile du fait que l'indice est indissociable du nom de la variable. Notre météorologue aura intérêt à utiliser un TABLEAU (que l'on appelle aussi VARIABLE INDICÉE). Un tableau est un ensemble de variables qui portent le même nom : ce sont les éléments du tableau. Pour les repérer, ils sont suivis de leur numéro d'ordre mis entre parenthèses : c'est l'indice.
dans notre exemple, on notera :
T(1)
T(2)
T(3)
.
.
.
T(31) s'il s'agit des températures du mois de juillet.
Le problème précédent est alors très simple à résoudre :
si N = 14 et si T(14) = 17 alors T(N) = 17 (Et mettez un pull pour aller au feu d'artifice).
Comme vous pouvez le constater, chaque élément du tableau est équivalent à une case numérotée qui s'utilise comme une variable ordinaire. La différence est que l'on peut "balayer" les éléments en faisant varier l'indice. Nous verrons des exemples dans la suite.

Règles d'utilisation des tableaux

L'indice doit être un nombre entier positif ou nul. Il est souvent plus commode de commencer par l'indice 1. Vous n'aurez normalement pas à vous soucier de la valeur maximum.
Cependant, pour des tableaux dont l'indice dépassera 1Ø, vous devrez indiquer à l'ordinateur qu'il doit réserver de la place en mémoire pour le tableau :

c'est le rôle de l'instruction DIM. La syntaxe est la suivante : avant toute utilisation du tableau, vous écrirez dans une ligne du programme DIM T(31).
Cela signifie que ATMOS réserve en mémoire une place pour un tableau dont l'indice ne dépassera pas 31. Si le tableau a 10 éléments au maximum cette réservation est inutile.
Enfin, pour donner un nom à vos tableaux, vous utiliserez les mêmes règles que pour les variables.

```
DIM TEMP (31)
DIM X3 (86)
DIM NOTE (27)
```

sont les déclarations de tableaux correctes. Rappelons quand même que seuls les deux premiers caractères sont pris en compte.

Remplissage d'un tableau

Après l'instruction DIM, le tableau existe mais tous ses élements sont nuls (comme pour les variables ordinaires).
Il faut bien sûr donner à ses éléments les valeurs voulues. Cela peut se faire de diverses façons :

— Par affectation.
Comme pour les variables, on écrira :

```
10  DIM T(31)
20  T(1) = 25 : T(2) = 27 : T(3) = 28 : T(4) = 24...
```

Vous avez naturellement trouvé vous-même l'inconvénient majeur : c'est très long. Imaginez un peu le pauvre météorologue qui devra entrer le tableau des températures pour une années.

— Par un INPUT.
Rappelez-vous l'instruction INPUT qui suspend le déroulement du programme et vous demande un nombre ou une chaîne qu'elle range ensuite dans une variable.
Vous pourrez l'utiliser pour les tableaux mais il faudra autant de INPUT que d'éléments. Ce qui nous amènera à une boucle qui permettra d'éviter les répétitions. Voici un exemple :

```
10  DIM T(31)
20  FOR I=1 TO 31
30  PRINT "Jour numéro";I;
40  INPUT T(I)
50  NEXT I
```

A l'exécution, le programme rappelle à l'utilisateur le numéro du jour dans lequel il doit donner la température. Ce dernier n'a qu'a taper la valeur, [RETURN] et recommencer 30 fois : c'est beaucoup plus rapide.

Le tableau est alors rempli et prêt à être utilisé.
Dans certains cas, vous ne saurez peut-être pas exactement le nombre d'éléments du tableau. C'est le cas ici si l'on généralise à d'autres mois que celui de juillet. On convient alors d'une valeur qui arrêtera l'entrée des données. Ce doit être une valeur normalement aberrante. Ici, par exemple 99 : si la température atteignait 99 °C, vous n'auriez vraisemblablement plus besoin d'ATMOS.
Vous écrirez le programme suivant :

```
10  DIM T(31)
20  REPEAT
25  I=I+1
30  PRINT "Jour numéro";I;
40  INPUT T(I)
50  UNTIL T(I)=99
```

Quand vous arrivez au dernier jour du mois, tapez 99 et vous sortirez de la boucle.

— Par READ et DATA.
Ces instructions seront détaillées au chapitre suivant : sachez seulement qu'elles sont équivalentes à des affectations multiples mais sous une forme beaucoup plus condensée.

Opérations sur les tableaux

Le tableau est rempli, on peut maintenant l'utiliser.

— *Affichage des éléments.*
Il suffit là encore d'une boucle. Exemple :

```
100  FOR I=1 TO 31
110  PRINT T(I);
120  NEXT I
```

— *Recherche de l'élément minimum.*
Pour le météorologue, il est intéressant de connaître la température minimale du mois. Il suffit de parcourir le tableau en faisant varier l'indice. Une variable TM sert à garder le minimum : Si l'élément est plus petit que TM, alors TM prend la valeur de l'élément sinon elle est inchangée. Voici le programme :

```
200  REM Recherche du minimum d'un tableau
210  TM=100
220  FOR I=1 TO 31
230  IF T(I)<TM THEN TM=T(I)
240  NEXT I
250  PRINT "La température minimale a été de"; TM; "Degrés"
```

Naturellement, sur 31 éléments vous irez aussi vite sans ATMOS mais imaginez un peu 1000 ou 5000 éléments !
Et ce petit programme nous servira plus tard dans ce que l'on appelle un tri.

```
5 REM TEMPERATURE
10 DIM T(31)
20 FOR I=1 TO 31
40 PRINT"JOUR NUMERO ";I;
45 INPUT T(I)
50 NEXT I
100 FOR I=1 TO 31
110 PRINTT(I);
120 NEXT I
200 REM RECHERCHE DU MINIMUM D'UN TABLEAU
210 TM=100
220 FOR I=1 TO 31
230 IF T(I) < T(M) THEN TM=T(I)
240 NEXT I
245 PRINT
250 PRINT "La temperature minimale";
260 PRINT" a ete de ";TM;" Degres"
```

Vous pourrez de la même façon rechercher le maximum ou l'égalité de deux éléments. Essayez par exemple d'écrire le programme qui vous donnera le jour ou les jours où la température était égale à 27 degrés.

— *Calcul d'une valeur moyenne.*
Tout le monde sait ce qu'est une moyenne : vitesse moyenne, consommation moyenne, note moyenne...
Les éléments d'un tableau sont en général de même nature et il est souvent intéressant d'en faire la moyenne. Et c'est très facile :

```
300  REM Calcul de la moyenne des éléments d'un tableau
310  SOM=0
320  INPUT "Nombre d'éléments du tableau T";N
330  FOR I=1 TO N
340  SOM=SOM+T(I)
350  NEXT I
360  PRINT "La moyenne est"; SOM/N
```

Ce programme peut s'appliquer à n'importe quel tableau ; vous pourrez l'appliquer à tous les exemples déjà cités.
Le voici, tel qu'il figure sur la cassette :

```
5 REM+++TEMPERATURE ET MOYENNE+++
6 REM
10 DIM T(31)
15 CLS:PAPER 6:INK 1
20 FOR I=1 TO 31
40 PRINT"JOUR NUMERO ";I;
```

```
45 INPUT T(I)
50 NEXT I
60 CLS:PRINT "Temperatures "
100 FOR I=1 TO 31
110 PRINT T(I);
120 NEXT I
130 PRINT:PRINT
200 REM REHERCHE DU MINIMUM D'UN TABLEAU
210 TM=100
220 FOR I=1 TO 31
230 IF T(I)<TM THEN TM=T(I)
240 NEXT I
250 PRINT"La temperature minimale ";
251 PRINT"a ete de ";TM;"degres"
300 REM CALCUL DE LA MOYENNE
301 REM DES ELEMENTS D'UN TABLEAU
310 SOM=0
320 PRINT:PRINT
330 FOR I=1 TO 31
340 SOM=SOM+T(I)
350 NEXT I
360 PRINT"La temperature moyenne ";
361 PRINT"est ";INT(SOM/31);" degres"
```

Autres exemples de tableaux

Tout ce qui se classe, se présente sous la forme d'une suite ou d'une liste pourra aisément être mis sous la forme d'un ou de plusieurs tableaux. Citons par exemple :

— Les notes d'un élève pour le professeur.

— Les articles du magasin pour le commerçant : il pourra utiliser un tableau pour les références, un pour les prix, un pour la quantité...

— Les résultats de mesures pour l'ingénieur qui pourra ensuite en faire un traitement mathématique pour calculer ce qui l'intéresse. Enfin les exemples seront encore plus nombreux avec les tableaux à plus d'un paramètre que nous allons voir maintenant.

Tableaux à deux dimensions et plus

Retour à la station météo : les mesures se font maintenant plusieurs fois par jour. Vous obtiendrez des résultats qui ressembleront à ceci :

Heure	Ø	6	12	18
Jour				
1	15	16	25	24
2	18	17	28	27
3	19	19	29	27
31	14	15	24	24

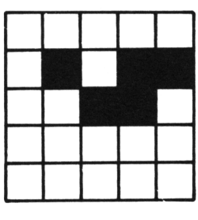
C'est vraiment ce que l'on appelle un tableau habituellement et c'est de là que vient le nom. On parle alors de tableau à deux paramètres ou à deux dimensions.
Vous l'écrirez sous la forme :
T(J,H) où J représentera l'indice du jour et H celui de l'heure. Ainsi, T(14,2) représentera la température du 14 juillet à 12 heures, si on convient d'affecter à H les valeurs Ø, 1, 2, 3. (Remarquons que nous avons utilisé ici pour H l'indice Ø pour avoir une correspondance simple entre l'heure et H : heure = 6*H)
Il aura fallu, au préalable faire une réservation de l'espace mémoire par DIM T(31,3) pour un mois de 31 jours et 4 mesures par jour.
Et si vous désirez des statistiques sur l'année, à vous le tableau à trois dimensions. T(M,J,H) contiendra les températures de tous les jours de l'année à différentes heures classés par mois. Exemple : T(3,5,2) vous donnera la température qu'il faisait le 5 mars à 6 heures du matin.
A partir de ce tableau, vous pouvez effectuer tous les calculs ou recherches du paragraphe précédent mais c'est un peu plus compliqué. Vous utiliserez des boucles imbriquées pour faire varier les trois indices.

```
5 REM TEMPERATURE 3
10 DIM T(12,31,4)
20 FOR M=1 TO 12
30 :   FOR J=1 TO 31
50 :       FOR H=0 TO 3
60 :       PRINT "MOIS";M
70 :       PRINT "JOUR";J
80 :       PRINT "HEURE";H*6
90 :       INPUT "TEMPERATURE";T(M,J,H)
100 :      NEXT H
110 :  NEXT J
195 NEXT M
```

RUN et bon courage ! Vous avez 4*31*12 soit 1488 valeurs à entrer.
Remarquez l'ordre des boucles : au départ M = 1 et J = 1, H prendra successivement les valeurs Ø, 1, 2, 3, ensuite J = 2 et on recommence...
Cela signifie que l'on donnera les 4 températures d'un jour puis celles du suivant et ainsi de suite pour tous les jours du mois ; après on change de mois.
Un petit inconvénient de ce programme : les mois ont toujours trente et un jours. On pourra bien sûr remplir les fins de mois avec des valeurs aberrantes du genre 99 pour distinguer les jours qui n'existent pas. Vous pouvez également remplacer la boucle des J par une boucle REPEAT UNTIL comme dans l'exemple précédent.
Lorsque le tableau sera rempli, vous pourrez faire des moyennes sur le mois, le jour, ou l'année, chercher le minimum ou le maximum ou bien encore trouver le jour ou l'écart de température est le plus grand.
A vous de jouer.



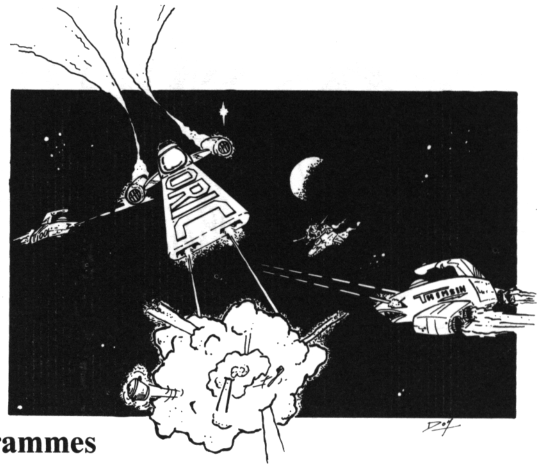
Un autre exemple : la bataille navale ou tout jeu qui utilise une grille de représentation. Vous utiliserez un tableau pour préciser l'état de chaque case.



Chaque case est repérée par son numéro de colonne C et celui de ligne L. Voici un exemple simple de bataille navale sur une grille 5 × 5 qui vous permettra de jouer avec ATMOS.

```
10 REM+++BATAILLE NAVALE+++
15 CLS:PAPER 5
20 DIM T(5,5)
30 T(2,2)=1:T(2,4)=1:T(3,3)=1
40 T(4,3)=1:T(5,2)=1
50 PRINT"QUELLE CASE VOULEZ-VOUS"
60 INPUT "COLONNE";C
70 INPUT"LIGNE";L
90 W=W+1
100 IF T(C,L)=1 THEN 120
110 PRINT"RIEN":GOTO 50
120 G=G+1
130 IF G=5 THEN 150
140 PRINT"TOUCHE";:GOTO 50
150 PRINT"COULE"
160 PRINT
170 PRINT"VOUS AVEZ GAGNE EN";W;"COUPS"
180 END
```

Vous devez maintenant être en mesure de réaliser le programme inverse, c'est-à-dire celui qui cherchera vos bateaux.



Les sous-programmes

Exemple

Supposons que nous ayons un programme qui effectue des calculs et que nous désirions qu'il affiche successivement les résultats en gros caractères clignotants pendant cinq secondes au milieu de l'écran.
Nous savons faire cela :

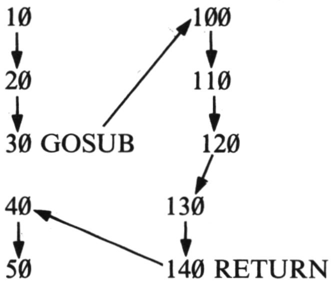
```
5 REM CLIGNOTEMENT
110 PLOT 11,7,12:PLOT 12,7,10
115 PLOT 13,7,STR$(X)
120 PLOT 11,8,12:PLOT 12,8,10
125 PLOT 13,8,STR$(X)
130 WAIT 500:CLS
```

Si vous avez oublié quelques subtilités de l'instruction PLOT ainsi que des caractères préfixés, revoyez le chapitre correspondant.
Deux remarques s'imposent :
Si vous faites plusieurs calculs vous devrez écrire à chaque fois ces trois lignes pour afficher un résultat. Bien sûr vous utiliserez l'éditeur qui vous permettra de les recopier mais vous allez perdre du temps et de la place en mémoire.
Ces trois lignes n'ont rien à voir avec le reste du programme et il serait peut-être bon de les mettre à part.
Dans ce cas, vous aurez intérêt à utiliser un sous-programme.
Un sous-programme est une partie de programme que l'on met à part soit parce qu'elle n'a rien à voir avec le reste (ce qui nuit à la lisibilité du programme), soit parce qu'on l'utilise plusieurs fois.
Reprenons notre exemple :
Pour que les trois lignes deviennent un sous-programme, il faudra signaler que ce travail-là est fini : on rajoutera à la fin
140 RETURN
Ce qui signifie retour à ce que l'on était en train de faire.
Le début du sous-programme peut commencer par n'importe quelle instruction. On conseille vivement d'indiquer dans un REM ce que fait le sous-programme en question. Nous ajouterons :
100 REM affichage clignotant double hauteur
Voyons maintenant comment aiguiller ATMOS vers le sous-programme. Prenons un exemple tout bête : un compteur. Nous allons afficher la suite des nombres entiers.
La méthode est évidente :
— on affiche le nombre X
— on l'augmente de 1
— on recommence
Il suffit de fixer au départ X à zéro et nous aurons réalisé le compteur. Pour afficher il faut envoyer au sous-programme. Cela se fait par l'instruction GOSUB N° de la ligne où commence le sous-programme.

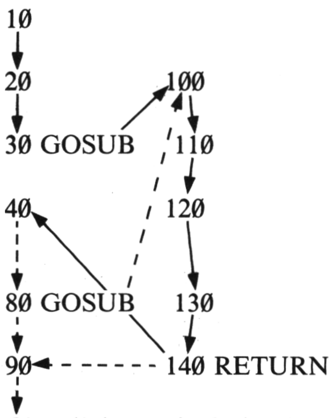
D'ou le programme complet :

```
5 REM compteur
10 CLS
20 X=0
30 GOSUB 100:REM Affichage
40 X=X+1
50 GOTO 30
60 END
100 REM Affichage double hauteur
110 PLOT 11,7,12:PLOT 12,7,10
115 PLOT 13,7,STR$(X)
120 PLOT 11,8,12:PLOT 12,8,10
125 PLOT 13,8,STR$(X)
130 WAIT 500:CLS
140 RETURN
```

Le déroulement du programme est alors le suivant :



On aurait pu avoir un deuxième calcul qui aurait nécessité un deuxième affichage donc un deuxième appel au sous-programme. Si par exemple en ligne 80 on avait un GOSUB 100, le RETURN ramènerait à la ligne suivante la ligne 80.



C'est là la particularité du GOSUB par rapport au GOTO : le GOSUB garde en mémoire le numéro de la ligne suivante qu'il restitue après le RETURN. C'est pourquoi on revient toujours à la suite normale du programme ; le sous-programme constitue en quelque sorte une parenthèse. Le programme normal est appelé programme principal. Il doit se terminer par un END.
L'utilisation systématique de sous-programme doit donner des programmes bien structurés donc plus faciles à comprendre. Elle est bien sûr vivement conseillée.

Passage de paramètres

Dans l'exemple précédent, nous avons passé un paramètre sans le savoir comme Monsieur Jourdain. Regardez bien le sous-programme : vous y verrez la variable X. Quelle est la valeur de X ? Elle est fixée par le programme principal. Ce dernier a passé au sous-programme un nombre qu'il lui demande d'afficher. Cela s'appelle "passer un paramètre".
Compliquons un tout petit peu le programme précédent. On voudrait aussi afficher en plus mais ailleurs sur l'écran le carré du même nombre.

On pense bien sûr à utiliser le même sous-programme. Mais notre sous-programme affiche en un endroit détermité de l'écran. Pour qu'il affiche là où on le désire, il est nécessaire de le modifier. Puisque les deux premiers paramètres de l'instruction PLOT représentent les numéros de la colonne et de la ligne, appelons-les C et L pour le dernier PLOT de la ligne 11Ø. Les valeurs des paramètres des autres PLOT s'en déduisent aisément et le sous-programme devient alors :

```
1ØØ  REM affichage n'importe où
11Ø  PLOT C-2, L,12: PLOT C-1,L,1Ø : PLOT C,L,STR$(X)
12Ø  PLOT C-2, L+1, 12: PLOT C-1, L+1, 1Ø: PLOT C, L+1,
     STR$ (X)
13Ø  WAIT 5ØØ: CLS
14Ø  RETURN
```

Ce sous-programme nécessite pour fonctionner le passage de trois paramètres : X nombre à afficher, L ligne où on affiche et C colonne du premier chiffre.
Dans le programme principal, avant le GOSUB, il faut affecter à ces trois variables les valeurs voulues.

```
1Ø  CLS
2Ø  N=Ø
3Ø  X=N: C=13: L=7: GOSUB 1ØØ: REM affichage du nombre
4Ø  X=N↑2: C=2Ø: L=15: GOSUB 1ØØ: REM affichage du carré en bas à
droite
5Ø  N=N+1
6Ø  GOTO 3Ø
```

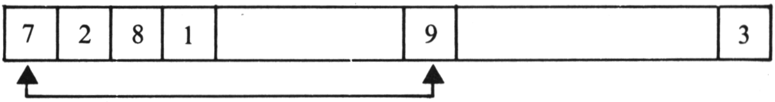
Comme vous pouvez le constater, le passage des paramètres est assez lourd et peu commode en BASIC. C'est un de ses défauts.

Voici le programme complet

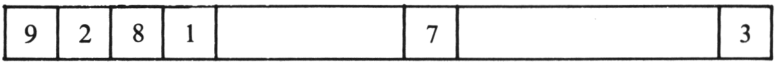
```
5 REM CLIGNOTEMENT LOCALISE
10 CLS
20 N=0
30 X=N:C=13:L=7:GOSUB 100
40 X=N*N:C=20:L=15:GOSUB 100
50 N=N+1
60 GOTO 30
70 END
99 REM
100 REM Affichage n'importe ou
110 PLOT C-2,L,12:PLOT C-1,L,10
115 PLOT C,L,STR$(X)
120 PLOT C-2,L+1,12:PLOT C-1,L+1,10
125 PLOT C,L+1,STR$(X)
130 WAIT 50:CLS
140 RETURN
```

Exemple d'utilisation de sous-programmes

Nous vous proposons, à titre d'exemple, un programme simple de tri. Trier des nombres, c'est les classer par ordre croissant ou décroissant. C'est très utile dans toutes les opérations de fichiers. Ce programme n'a aucune prétention utilitaire, (il sera très lent à l'exécution mais il est bien structuré et il devrait vous faire comprendre aisément la méthode utilisée).
Le principe est le suivant :
Nous avons N nombres à trier. On cherche le plus grand (Premier sous-programme) et on l'échange avec le premier. (Deuxième sous-programme).



On a alors la disposition suivante :



Il suffit ensuite de recommencer avec le reste et ceci jusqu'au dernier nombre.

```
10 REM+++TRI DE NOMBRES+++
11 REM+++PAR ORDRE DECROISSANT++
15 CLS:PAPER3:INK4
20 PRINT"COMBIEN AVEZ-VOUS DE NOMBRES A TRIER"
21 INPUT N
25 DIM T(N)
3 FOR I = 1 TO N:REM remplissage tableau
40 INPUT"DONNEZ UN NOMBRE";T(I)
50 NEXT I
60 REM progr. ppal du tri
70 FOR I=1 TO N-1
80 GOSUB 200:REM recherche
81 REM                du rang maximum
90 GOSUB 300 :REM permutation du Ieme
91 REM                et du maximum
100 NEXT I
110 REM affichage nombres a trier
120 FOR I=1 TO N
130 PRINT T(I);
140 NEXT I
150 END
200 REM sous-programme de recherche
201 REM du rang maximum
210 MAX=T(I):M=I
220 FOR K=I TO N
230 IF T(K)>MAX THEN MAX=T(K):M=K
240 NEXT K
250 RETURN
300 REM sous programme de permutation
```

```
301 REM du Ieme element et du maximum
310 AUX=T(I):T(I)=T(M):T(M)=AUX
320 RETURN
```

Le programme principal du tri proprement dit est très succint (lignes 7Ø à 1ØØ). Auparavant il a fallu remplir le tableau. Ensuite (lignes 12Ø à 14Ø) on affiche les nombres triés.
Le premier sous-programme renvoie dans M le rang du maximum des éléments non triés.
Le deuxième permute les éléments T(I) et T(M) du tableau. Il utilise une variable AUX indispensable pour ne pas perdre un des éléments.
Le END de la ligne 15Ø est ici indispensable. S'il n'existait, cela entraînerait un message d'erreur "RETURN WITHOUT GOSUB".

Les fonctions mathématiques

Fonctions préprogrammées

Ce sont celles que vous avez l'habitude de trouver sur les calculatrices scientifiques.

Fonctions trigonométriques :
SIN (X), COS (X), TAN (X)
N'oubliez pas les parenthèses et n'oubliez pas que X est en Radians. π Radians = 180 degrés. Le nombre π est préprogrammé : il s'appelle PI. Essayez donc
?SIN (PI) et
?COS (PI/2)...
Vous avez à votre disposition une fonction trigonométrique inverse : l'arctangente appelée ici ATN(Y)
Exemple : ?ATN(3) Attention elle vous donne l'angle en Radians.

Fonctions logarithmes et exponentielle.
Elles s'écrivent :
LOG(X) pour le logarithme décimal.
LN(X) pour le logarithme népérien.
EXP(X) pour l'exponentielle.
Nous n'aurons pas la prétention de vous faire un cours sur la question.
Rappelez-vous toujours que ce sont des fonctions et pas des instructions. Elles ne peuvent figurer qu'après une instruction ou un signe =.

Fonctions définies par l'utilisateur

ATMOS est bien sûr plus fort qu'une calculatrice. Si dans un calcul, vous utilisez souvent une même fonction vous pouvez lui donner un nom. C'est d'autant plus intéressant qu'elle est plus compliquée.
Voici un exemple : j'ai besoin de calculer plusieurs fois la fonction homographique

$$f(x) = \frac{2x + 3}{7x - 4}$$

Avant l'utilisation en début de programme vous écrirez :
3Ø DEF FN F(X) = (2*X+3)/(7*X−4)
Essayez pour vérifier :
4Ø PRINT FNF(2)
A l'exécution ATMOS vous répondra : Ø.7 ce qui est bien la valeur de la fonction pour X=2. Dans la suite du programme vous pourrez utiliser cette fonction autant de fois que vous le voudrez.
Notez bien que les fonctions définies par DEF FN jouent un peu le même rôle que les sous-programmes. Elles vous éviteront des répétitions inutiles et rendront les programmes plus simples à comprendre.
Attention aux différences : les fonctions ne peuvent pas être mises directement dans une ligne de programme : ce ne sont pas des instructions. D'autre part, la définition de la fonction doit figurer obligatoirement avant l'utilisation.

Retour sur les opérateurs logiques

Rappelez-vous les sauts conditionnels : IF... THEN...
Après le THEN vous écrirez une ou plusieurs instructions. Après le IF doit figurer une condition que l'on appelle plus savamment un BOOLEEN. Un Booleen ne peut prendre que deux valeurs : VRAI ou FAUX. On l'obtient généralement en comparant deux grandeurs, soit deux variables soit une variable et une constante. Rappelons que si la condition est vraie le programme exécute les instructions qui se trouvent après le THEN, si elle est fausse il exécute celles après le ELSE s'il existe ou passe à la ligne suivante s'il n'existe pas.
Les opérateurs de comparaison sont ceux des mathématiques avec quelques modifications dans les notations.
= égal
< > différent de
> supérieur
< inférieur
> = supérieur ou égal
< = inférieur ou égal
Vous pouvez combiner deux ou plusieurs conditions par des opérateurs logiques. Il n'en existe que deux : OR et AND.
Si p et q sont deux Booléens ;
p AND q est vrai seulement si p est vrai et q est vrai. Dans tous les autres cas p AND q est faux.
p OR q est vrai si p ou q ou les deux sont vrais. p OR q est faux uniquement lorsque les deux sont faux.
Exemples :
IF X>3 AND X<9 THEN...
Équivaut à l'expression mathématique : si $3<X<9$ alors...
IF X>3 OR X<9 THEN...
Ne présente aucun intérêt puisque la condition est toujours vraie.
Ces opérateurs sont très utiles dans les analyses de réponses ou de résultats pour aiguiller l'interlocuteur vers différentes possibilités.
Vous rencontrerez de nombreux exemples dans les chapitres suivants.

CHECK-LIST

DIM 212
GOSUB 193
SIN 171
COS 171
TAN 172
ATN 172
LOG 174
LN 173
EXP 173
DEF FN 177
AND 198
OR 198

TRAINING

• Mettre sous forme de tableaux les références des articles d'un catalogue : les prix et les quantités en stock. Écrire le programme permettant de trouver un article connaissant sa référence.
• A partir du tableau des températures, déterminer la température moyenne pour un mois donné à une heure donnée. Vous pourrez, dans un premier temps, supposer que tous les mois ont trente et un jours et envisager ensuite le cas réel.
• Programme de bataille navale sur grille 2Ø × 2Ø avec plusieurs types de bateaux occupant 1 à 4 cases.
• Programme de bataille navale complet où ATMOS cherche vos bateaux.
• Et pourquoi pas une bataille dans l'espace ?



Traitement des données et complément sur les chaînes de caractères

Un programme devient réellement intéressant lorsqu'il permet d'effectuer des calculs sur des séries de données.
Ces informations, vous pouvez les lui fournir au fur et à mesure qu'il les demande. Comme dans l'exemple qui suit :

```
10  INPUT A,B
20  C=(A+B)/2
30  PRINT "A="; A, "B="; B, "C="; C
RUN
? 2,3
A=2 B=3 C=2,5
Ready
```

A ce niveau, si vous voulez refaire le calcul pour des valeurs différentes de A et B deux solutions sont possibles :
• soit revenir à l'instruction 10 à l'aide d'un GOTO, mais ATMOS s'arrêtera sur le ? de l'INPUT ;
• soit fournir au programme une série de données dès l'écriture de celui-ci.
Cela se fait par deux instructions du BASIC :
— DATA qui est la description des données
— READ qui est la lecture de celles-ci.

Des données incorporées

Les données que les instructions DATA permettent d'incorporer au programme sont des constantes. Elles peuvent être soit des nombres soit des mots :

```
10  DATA I,4,6,7... 89,678,78965
20  DATA JANVIER, FÉVRIER, MARS, AVRIL
```

Notez que ces différentes données sont séparées par des virgules.
Cette séparation est nécessaire pour que l'ordinateur distingue chaque donnée.
Sachez aussi que l'instruction DATA est le moyen le plus simple et pratique de conserver des données sur *cassette*.
Supposons que nous voulions afficher les jours de la semaine.

```
10  REM SEMAINE
15  CLS
20  DATA LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI,
SAMEDI, DIMANCHE
30  READ J1$, J2$, J3$, J4$, J5$, J6$, J7$
40  PRINT J1$, J2$, J3$, J4$, J5$, J6$, J7$
50  END
```

L'exécution de READ en ligne 30, associe la variable J1$ à la première constante LUNDI, puis la variable suivante à la constante suivante et ainsi de suite jusqu'à l'épuisement des constantes et des variables. Vous aurez sans doute remarqué que ce programme n'est pas économique puisqu'on écrit 14 fois les variables "jours". Une boucle FOR... NEXT s'impose pour cette lecture répétitive. Le programme devient, à partir de la ligne 30 :

```
...
30  FOR I=1 TO 7
40  READ J$(I)
50  PRINT J$(I)
60  NEXT I
70  END
RUN
LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI,
DIMANCHE
Ready
```

Les instructions DATA dépendant d'un READ peuvent être placées n'importe où dans le programme. Elles peuvent contenir autant de constantes que la ligne le permet. Il n'y a pas de limite (hormis les problèmes de place en mémoire) au nombre des instructions DATA d'un programme.
Bien entendu, les types de données (numériques ou chaînes de caractères) doivent s'accorder avec les types de variables auxquelles elles vont être affectées.

Ainsi, le programme suivant :

```
10  READ A,B
20  PRINT A,B
30  DATA BONJOUR, TOTO
RUN
TM ERROR
```

sort en erreur TYPE MISMATCH car on a voulu affecter une chaîne de caractères (BONJOUR) à une variable numérique (A). Il convient de remplacer les variables A et B par des variables chaînes A et B pour sortir de l'erreur.
Dans le cas où les chaînes de caractères contiennent une virgule, des points ou des espaces significatifs en début ou en fin, elles doivent être encadrées par des guillemets :

```
100  DATA "LUNDI:", "MARDI :", ...
```

Une fois lues, les DATA ne sont pas utilisables dans la suite du programme. Si on essaye de les réutiliser, un message d'erreur signale : O.D. ERROR (OUT OF DATA) c'est-à-dire "il n'y a plus de données".
Si vous voulez que l'instruction READ vienne une seconde fois ou plusieurs fois relire les mêmes DATA, il vous faut utiliser l'instruction RESTORE.

Cette instruction fait remonter le pointeur associé aux instructions DATA:

```
10  CLS
20  DATA PIERRE, PAUL, MARIE, JACQUES, GÉRARD, SERGE
30  READ A$, B$, C$
40  RESTORE
50  READ D$, E$
60  PRINT A$;B$;C$;D$;E$
RUN
PIERRE PAUL MARIE PIERRE PAUL
```

READ D$, E$ reprend la lecture au début de la ligne 10.

Dans le cas où vous faites reprendre l'exécution d'un programme sans effectuer un "RUN", il est nécessaire de remettre toutes les variables numériques à zéro et les variables chaînes (dites "alphanumériques") à vide. Pour cela, il existe une instruction CLEAR à insérer juste après la ligne 10 du programme précédant :

```
10  CLS
15  CLEAR
20  DATA
...
```

Bien entendu, avant d'intégrer CLEAR dans n'importe quel programme, vérifiez que les valeurs calculées dans les instructions qui précèdent ne sont pas utilisées après cette instruction !



Un petit carnet d'adresses

Le nombre de vos amis et relations augmente ? Vous avez un problème de stockage de noms, adresses, numéros de téléphone ? Ce carnet électronique peut vous rendre service.
Que doit faire le programme ?
— demander à l'utilisateur le nom à chercher,
— afficher l'adresse et le numéro de téléphone correspondant,
— conserver en DATA la liste des noms, adresses, téléphones.

```
 1Ø  REM CARNET D'ADRESSES
 2Ø  CLS
 3Ø  INPUT "QUEL NOM DÉSIREZ-VOUS"; NOM$
 4Ø  READ N$, AD$, TEL$
 5Ø  IF N$ <>NOM$ THEN 4Ø
 6Ø  PRINT N$, AD$, TEL$
 7Ø  '
1ØØ  DATA CARPENTIER,34 RUE LALANDE,456 67 89
11Ø  DATA BERTRAND,1BVD JOFFRE,789 45 25
12Ø  DATA DUTILLEUL,67 PL LE VISTE,789 34 23
```

Lorsque l'utilisateur tape le nom cherché, le programme (ligne 5Ø) compare ce nom avec les noms de sa liste. Quand il trouve le même, il l'affiche avec l'adresse et le téléphone.
Essayez ce programme. S'il sort en erreur, malgré un nom correct, vérifiez que les "data" correspondants soient correctement entrés (pas de blanc avant ou après la virgule).
Qu'arrive-t-il si vous donnez un nom absent de la liste ou mal orthographié ? La liste est parcourue et l'ordinateur cherche encore une donnée après le dernier nom ce qui provoque une sortie en erreur : OD ERROR. On peut y remédier en y ajoutant les lignes suivantes :

```
35  C=O
...
45  IF N$="+++"THEN 7Ø
...
55  C=1
...
7Ø  IF C>Ø THEN 9Ø
...
8Ø  PRINT NOM$; "n'est pas dans le carnet"
...
9Ø  END
...
13Ø  DATA +++, +++, +++,
...
```

```
10 REM+++CARNET D'ADRESSES+++
11 REM
20 CLS:PAPER7:INK5
30 INPUT"Quel nom desirez-vous ";NOM$
35 C=0
40 READ N$,AD$,TEL$
45 IF N$="+++" THEN 70
50 IF N$<>NOM$ THEN 40
55 C=1
60 PRINT N$,AD$,TEL$
70 IF C>0 THEN 90
80 PRINT NOM$;" n'est pas dans le carnet"
90 END
100 DATA CARPENTIER,34 rue LALANDE,456 67 89
110 DATA BERTRAND,1 Bvd JOFFRE,789 34 23
120 DATA DUTILLEUL,67 Pl LE VISTE,432 23 04
130 DATA +++,+++,+++
```

La variable C nous sert à indiquer qu'on a trouvé le nom dans la liste : C vaut 1 à ce moment-là et en ligne 7Ø l'instruction renvoie en fin d'exécution.
La ligne 45 est une donnée fictive servant à marquer la fin des données de la liste. Lorsque l'ordinateur le lit, c'est qu'il a parcouru toute la liste sans trouver le nom cherché. Au lieu de chercher une autre donnée et de sortir en erreur, il va en ligne 7Ø où, comme C vaut Ø, il passe à la ligne suivante et affiche le message Untel n'est pas dans le carnet.
Vous pourrez augmenter à volonté la liste des DATA, à condition de rétablir en fin de liste la donnée fictive. Vous pouvez aussi améliorer ce programme et l'adapter à vos besoins.

Le jeu du cadavre exquis

"Le cadavre exquis boira le vin nouveau". Vous connaissez, peut-être ce jeu surréaliste qui consiste à mettre bout à bout des morceaux de phrases en comptant sur le hasard pour coller les morceaux. La fonction de calcul aléatoire RND ainsi que les propriétés du tableau de chaînes combiné avec des DATA vont nous aider à simuler ce jeu aussi inépuisable que cocasse.

- Les données : 39 éléments (13 noms, 13 verbes, 13 adverbes).
- Les résultats : obtenir des combinaisons aléatoires des trois catégories.

```
10 REM+++CADAVRE EXQUIS+++
11 REM
12 CLS
20 PRINT
30 PRINT"Ce programme produit des phrases";
31 PRINT" en francais aleatoire"
40 PRINT
50 PRINT"Au point d'interrogation repondez";
51 PRINT" par 'OUI' pour une autre phrase"
60 PRINT"'N' pour arreter"
```

```
70 PRINT:PRINT
80 PRINT"Voici la Premiere Phrase"
90 DIM A$(40)
100 FOR I=1 TO 39
120 READ A$(I)
130 NEXT I
140 PRINT A$(INT(13*RND(I)+1));" ";
150 PRINT A$(INT(13*RND(I)+14));" ";
160 PRINT A$(INT(13*RND(I)+27)):PRINT
165 INPUT R$:IF R$="OUI" THEN 140 ELSE 300
199 REM
200 DATA "LE CADAVRE","LE CHAT"
205 DATA "LA BELETTE","LA SOURCE"
210 DATA "LA BELLE CAPTIVE","LE COUREUR"
215 DATA "LE VIN","LA CUILLERE"
220 DATA "L'OISEAU","L'ELEPHANT"
221 DATA "LA CHOUETTE","LE PRESIDENT"
230 DATA "LE BERCEAU","DEVORE","SUCE"

235 DATA "HULULE","EXTRAPOLE","GRATTE"
240 DATA "SIFFLE","EMBRASSE","BATIFOLE"
245 DATA "SINGE","ETERNUE","S'ENERVE"
250 DATA "ANALYSE","HURLE","FOLLEMENT"
255 DATA "GOULUMENT","TENDREMENT"
260 DATA "SALEMENT","DELICATEMENT"
265 DATA "SINCEREMENT","SUBTILEMENT"
270 DATA "SAUVAGEMENT","PEU","BEAUCOUP"

275 DATA "BIZARREMENT","BIGREMENT"
280 DATA "FORT"
300 PRINT "Quand vous serez sature, "
301 PRINT "renouvelez mes donnees":END
```

Tableaux de chaînes

Les DATA c'est très pratique. A condition de ne pas vouloir utiliser les données en même temps. Pour les trier, par exemple, ou les lire aléatoirement. En effet, chaque fois, il faut relire la liste des DATA jusqu'à celle qui nous intéresse.

Mais vous savez depuis le chapitre précédent qu'il existe une autre façon de stocker (provisoirement) les données pour ensuite les traiter : *les tableaux,* suites ordonnées d'éléments. Voici un complément au chapitre précédent : *les tableaux de chaînes.*

Comment les identifier ? Seule la variable indiquant le nom du tableau change par rapport aux tableaux numériques. L'instruction de déclaration reste la même :

DIM JOUR$(7)

Comme vous le savez, l'instruction DIM réserve, en mémoire, la place néces-

saire pour ranger les éléments du tableau. Le chiffre 7 correspond aux sept éléments que constituent les jours de la semaine :

```
10 REM SEMAINE
15 INK 2:PAPER 0 :CLS
20 DIM JOUR$(7)
30 FOR I=1 TO 7
40 READ J$
50 JOUR$(I)=J$ ———
60 PRINT JOUR$(I)
70 NEXTI


100  DATA LUNDI, MARDI, MERCREDI, SAMEDI, DIMANCHE
```

Vous observez ici une combinaison DATA/TABLEAUX DE CHAÎNES. Que se passe-t-il ? Les données en DATA sont rangées dans un tableau qui, nous le verrons plus loin, facilitera leur exploitation.

Un tournage "au ralenti" du programme vous en éclaircira la logique :

• A partir de la ligne 30, au premier passage de la boucle : I vaut 1, l'instruction READ affecte la variable J$ de la première constante en DATA "LUNDI".

• En ligne 50, I valant toujours 1, le pointeur de l'ATMOS est à hauteur de la case 1 du tableau, laquelle reçoit J, c'est-à-dire "LUNDI".

• En ligne 60, ATMOS affiche J, "LUNDI". De plus, à cause du ";" le curseur reste sur la même ligne, en attendant l'instruction d'affichage suivante.

• En ligne 70, NEXT renvoie au FOR de la ligne 30. Cette fois I vaut 2, J reçoit la donnée suivante "MARDI", etc. jusqu'à 7.

Quel est l'intérêt du tableau ? De pouvoir accéder au données dans n'importe quel ordre. Exemple :

```
?JOUR$(5)
VENDREDI
?JOUR$(2)
MARDI
```

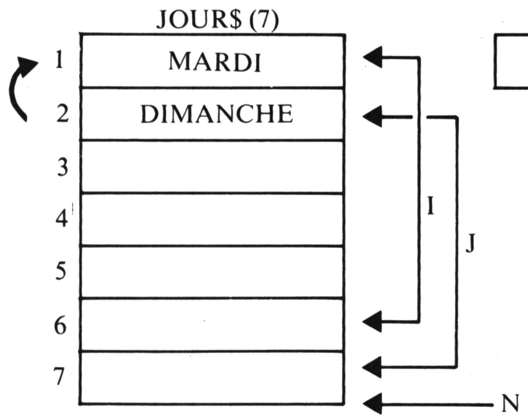
Ou encore, de se livrer à un *accès aléatoire :*

```
75 A=INT(7*RND(1)+1)
76 PRINTJOUR$(A)
```

Ou enfin d'effectuer un *tri alphabétique* des données :

```
78 N=7
80 FOR I=1 TO N-1
81 FOR J=I+1 TO N
83 IF JOUR$(I)>JOUR$(J) THEN 84 ELSE 85
84 Z$=JOUR$(I):JOUR$(I)=JOUR$(J):JOUR$(J)=Z$
85 NEXT J:NEXT I
86 FOR I=1 TO 7
87 PRINTJOUR$(I)
88 NEXT I
```

Commentaires :
Le principe de ce tri est une comparaison 2 à 2 des variables dans l'ordre croissant du tableau. Il y a permutation de la plus "petite" (dans l'ordre alphabétique) avec la plus "grande". La plus "petite" passe dans la position la plus basse, c'est-à-dire celle qui précède. Une variable intermédiaire est utilisée pour permettre le déplacement.

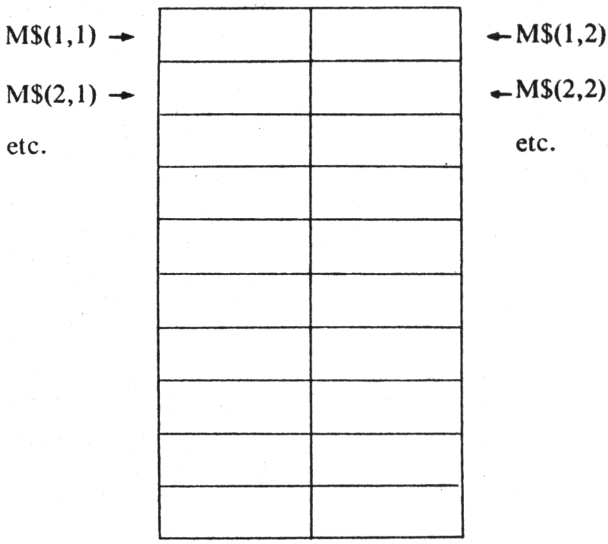


[78] Initiation du pointeur de fin de tableau.
[80] Première boucle du pointeur I. Elle s'arrête à l'avant-dernière case pour permettre le test avec le contenu de la suivante.
[81] Deuxième boucle du pointeur J. Elle commence à la case qui suit I pour permettre les comparaisons successives 2 à 2.
[83] Test du contenu des variables. La comparaison entre les codes ASCII des caractères des deux chaînes JOUR$(I) et JOUR$(J) fournit l'ordre alphabétique.
[84] Permutation. Z$ est la variable intermédiaire qui reçoit le contenu de la position "inférieure" en cas de permutation avec celui de la position "supérieure".
[86-87] Affichage du nouveau tableau rangé dans l'ordre alphabétique.

Tableaux de chaînes à deux dimensions

Plus généralement, on utilise des tableaux à 2 ou plusieurs dimensions :
DIM M$(1Ø,2)
Cette instruction déclare un tableau à 2 dimensions dont la première colonne varie de 1 à 1Ø et la seconde également.

En d'autres termes, elle réserve un tableau de 1Ø lignes et 2 colonnes :



Ce type de grille est utilisable pour toute sorte de jeux (nous vous avions proposés une bataille navale p. 91) mais aussi pour la gestion ou la révision de vos cours.
Si vous avez du mal à retenir votre vocabulaire anglais ou si vous désirez aider vos proches à retenir leur leçon, ATMOS peut se transformer en un répétiteur sympathique. Il suffira d'adapter les DATA du programme suivant à vos besoins.

Conception d'un petit répétiteur de leçon :
Vous désirez :
— créer un lexique bilingue,
— interroger l'élève,
— tester sa réponse,
— lui donner une seconde chance,
— afficher son résultat en fin d'interrogation.
Commençons par déclarer notre tableau à deux dimensions (cf. schéma ci-dessus).

```
1Ø  REM+ + +REPETITEUR1 + + +
2Ø  CLEAR
3Ø  CLS
4Ø  DIM M$(1Ø,2)
```

Créons la boucle de remplissage par les 1Ø mots en DATA et leur traduction :

```
5Ø  FOR I=1 TO 1Ø
6Ø  READ M$(I,1), M$(I,2)
7Ø  NEXT I
```

M$(1,1)	TO GET	OBTENIR	M$ (1,2)
2	TO PRINT	IMPRIMER	
3	⋮	⋮	
4	⋮	⋮	
5	⋮	⋮	
6	⋮	⋮	
7	⋮	⋮	
8	⋮	⋮	
9	LEFT	GAUCHE	
1Ø	RIGHT	DROITE	

Maintenant que le tableau est rempli, créons le module d'interrogation :

```
75 RJ=0:RF=0:REM Cpteurs reponses
76 REM             justes et fausse
80 FOR I=1 TO ND :REM boucle d'interrogation
90 C=0:REM Initial.cpteur questions
100 PRINT M$(I,1):REM Question
120 INPUT R$:REM Reponse
125 REM Test de validite de la reponse
130 IF R$=M$(I,2) THEN 175
140 C=C+1:IF C > 2 THEN 170
160 PRINT"RECOMMENCEZ":GOTO 100
170 PRINT"C'ETAIT ";M$(I,2):RF=RF+1:GOTO 180
175 PRINT"BRAVO":RJ=RJ+1
180 NEXT I
```

Et enfin les données :

```
600 DATA 10,TO GET,OBTENIR,TO PRINT,IMPRIMER

605 DATA TO CLEAR,ECLAIRCIR,THE END,LA FIN
610 DATA NEW,NOUVEAU,TO LIST,LISTER

615 DATA TO SAVE,SAUVEGARDER,TO LOAD,CHARGER
620 DATA LEFT,GAUCHE,RIGHT,DROITE
630 RESTORE
999 END
```

On peut perfectionner ce programme en laissant à l'utilisateur le choix du nombre de DATA. Ceci revient à concevoir un tableau à 2 dimensions mais de longueur variable. Cette longueur sera déterminée par le nombre de DATA, d'où la nécessité de l'indiquer en début de liste des DATA afin de la faire lire par un READ et de l'affecter à une variable ND, dimensionnant le tableau. Ce qui entraîne les modifications suivantes :

— lecture du nombre de DATA :
  35 READ ND
— déclaration du tableau avec indice de longueur variable :
  4Ø DIM M$(ND,2)
— boucle de remplissage de longueur variable :
  5Ø FOR I=I TO ND
— boucle d'interrogation de longueur variable :
  8Ø FOR I=I TO ND
— aménagement des DATA
  6ØØ DATA 1Ø, TO GET, OBTENIR etc. On pourrait écrire DATA 3Ø à condition de rentrer une liste de 3Ø mots à traduire, soit 6Ø mots en DATA.

Dernières améliorations :

— Test de continuation de l'interrogation pour l'utilisateur :
  19Ø PRINT: PRINT: "Voulez-vous continuer?(O/N)": INPUT RE$
  192 IF RE="O" THEN IØ ELSE END
— un titre et quelques explications pour "l'élève". Ce qui donne, en définitive :

```
10 REM+++REPETITEUR1+++
20 CLEAR
30 CLS:PAPER4:INK1
31 PRINT TAB(20)"REPETITEUR"
32 PRINT :PRINT"Traduisez en Francais:"
35 READ ND
40 DIM M$(ND,2)
50 FOR I=1 TO ND:REM boucle remplissage tableau
60 READ M$(I,1),M$(I,2)
70 NEXT I
75 RJ=0:RF=0:REM CPteurs reponses
76 REM           Justes et fausse
80 FOR I=1 TO ND :REM boucle d'interrogation
90 C=0:REM Initial.cPteur questions
100 PRINT M$(I,1):REM Question
120 INPUT R$:REM Reponse
125 REM Test de validite de la reponse
130 IF R$=M$(I,2) THEN 175
140 C=C+1:IF C > 2 THEN 170
160 PRINT"RECOMMENCEZ":GOTO 100
170 PRINT"C'ETAIT ";M$(I,2):RF=RF+1:GOTO 180
175 PRINT"BRAVO":RJ=RJ+1
180 NEXT I
185 CLS:PRINT"REPONSES JUSTES "RJ
```

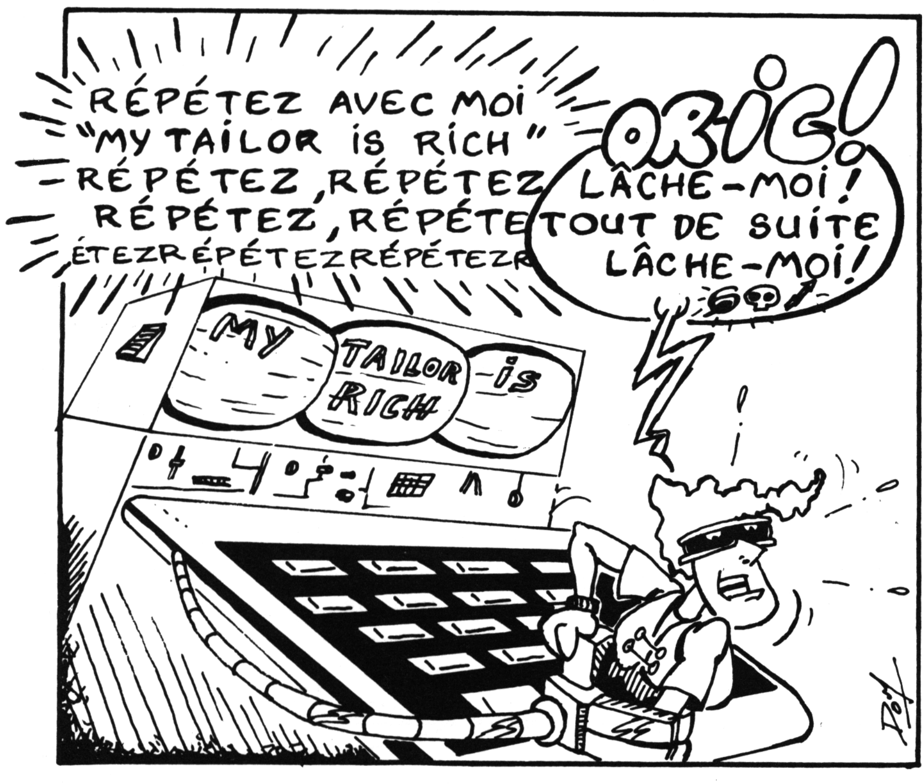
```
190 PRINT:PRINT"Voulez-vous continuer(O/N)";
191 INPUT RE$
192 IF RE$="O" THEN 10 ELSE 999
600 DATA 10,TO GET,OBTENIR,TO PRINT,IMPRIMER

605 DATA TO CLEAR,ECLAIRCIR,THE END,LA FIN
610 DATA NEW,NOUVEAU,TO LIST,LISTER

615 DATA TO SAVE,SAUVEGARDER,TO LOAD,CHARGER
620 DATA LEFT,GAUCHE,RIGHT,DROITE
630 RESTORE
999 END
```

Un menu pour votre répétiteur (ON...GOTO)

Pourquoi ne pas proposer à l'utilisateur diverses activités, telles que la traduction du français à l'anglais (ou tout autre langue), la révision des mots ? Cela nécessite une réorganisation du programme. Des instructions nouvelles vont nous y aider.
Que faut-il faire ? D'abord offrir un menu avec aiguillage selon le choix : ON... GOTO est l'instruction qu'il nous faut. Ensuite restructurer notre programme avec l'aide de GOSUB, que vous connaissez.



1) L'affichage du menu :
Il s'agit de composer une première "page écran", à l'aide de PRINT TAB, qui aura la forme suivante :

```
RÉPÉTITEUR
1 — ANGLAIS/FRANÇAIS
2 — FRANÇAIS/ANGLAIS
3 — RÉVISION
4 — FIN DE TRAVAIL
VOTRE CHOIX ?
```

Application :

```
10 REM+++REPETITEUR2+++
11 REM-----------------
20 CLEAR
30 CLS:PAPER0:INK3
31 PRINT TAB(20)"REPETITEUR"
32 PRINT:PRINT TAB(20)"1-Anglais/Francais"
33 PRINT:PRINT TAB(20)"2-Francais/Anglais"
34 PRINT:PRINT TAB(20)"3-Revision"
35 PRINT:PRINT TAB(20)"4-Fin de travail"
36 PRINT:PRINT:PRINT TAB(25)"Votre choix";
37 INPUT R$
```

2) L'aiguillage selon le choix :
• Utilisons pour cela, l'instruction ON X GOTO Y. X est le numéro du choix, Y le numéro de la ligne où débute la séquence choisie :

```
38 R=VAL(R$)
39 IF ASC(R$)<49 AND ASC(R$)>53 THEN 36
40 ON R GOTO 45,200,300,400
```

Si R vaut 1, le programme ira en 45, si R vaut 3 en 3ØØ, etc.

3) Restructuration du programme :
il faut ajouter au programme les parties correspondant aux choix 2,3,4. Cela entraîne une restructuration de l'ensemble car des séquences vont être en commun pour ces trois choix, comme la déclaration et le chargement du tableau. Le recours au sous-programme avec l'instruction GOSUB s'impose. Le nouveau programme s'organisera de la façon suivante :

• 1 programme principal en 3 modules
  - Angl./Fran.
  - Fran./Angl.
  - Révision

• 2 sous-programmes : remplissage du tableau / affichage du bilan

Le module anglais/français existe déjà, il faut simplement y inclure les appels de sous-programmes :

```
45  REM MODULE ANGLAIS/FRANÇAIS
5Ø  GOSUB 5ØØ
75  RJ=Ø:RF=Ø
...
19Ø  GOSUB 55Ø
195  GOTO 2Ø
```

Le module français/anglais étant l'inverse du précédent, il s'agira d'inverser la lecture du tableau :

```
200 REM FRANCAIS/ANGLAIS
201 REM------------------
205 CLS:PAPER 4:INK1
210 GOSUB 500
220 RJ=0:RF=0
225 FOR I=1 TO ND
230 C=0
235 PRINT M$(I,2)
240 INPUT R$
250 IF R$=M$(I,1) THEN 290
255 C=C+1:IF C>2 THEN 265
260 PRINT"RECOMMENCE":GOTO 235
265 PRINT"C'ETAIT: ";M$(I,1):RF=RF+1:GOTO 180
290 PRINT"BRAVO":RJ=RJ+1
295 NEXT I
296 GOSUB 550
298 GOTO 20
299 REM
```

Le module révision sera un simple affichage du tableau :

```
300 REM REVISION
301 REM------------------
305 CLS:PAPER6:INK0
310 GOSUB 500
320 FOR I=1 TO ND
325 PRINT M$(I,1),M$(I,2)
330 NEXT I
335 INPUT"RETOUR AU MENU(O/N)";R$
340 IF R$="O"THEN 20 ELSE 400
```

Le module de fin de travail, un simple envoi en fin de programme

```
399 REM
400 REM FIN DU TRAVAIL
401 REM------------------
405 GOTO 999
499 REM
```

Le sous-programme de déclaration et remplissage du tableau comportera

l'instruction de renvoi à l'instruction qui suit immédiatement l'appel (RETURN)

```
500 REM SOUS-PROGRAMME DE REMPLISSAGE
501 REM------------------
505 READ ND
510 DIM M$(ND,2)
515 FOR I=1 TO ND
520 READ M$(I,1),M$(I,2)
525 NEXT I
530 RETURN
549 REM
```

Le sous-programme de bilan, n'omettra pas l'effacement de l'écran pour préparer la troisième page-écran de ce programme et un test de continuation :

```
550 REM BILAN
551 REM------------------
555 CLS:PRINT"REPONSE(S) JUSTE(S): "RJ
560 PRINT:PRINT"REPONSE(S) FAUSSE(S): "RF
565 INPUT"ON #ONTINUE(O/N)";R$
570 IF R$="O" THEN 580 ELSE 400
580 GOTO 20
```

De la place, encore de la place

ATMOS a une grosse mémoire, vous le savez. Mais les chaînes de caractères en prennent beaucoup. Pour savoir où vous en êtes vous disposez de la fonction FRE(Ø) qui vous donne le nombre d'octets encore disponible dans la mémoire vive. Après avoir rentré le programme qui suit, tapez : ?FRE(Ø).

```
10 REM+++REPETITEUR2+++
11 REM----------------
20 CLEAR
30 CLS
31 PRINT TAB(20)"REPETITEUR"
32 PRINT:PRINT TAB(20)"1-Anglais/Francais"
33 PRINT:PRINT TAB(20)"2-Francais/Anglais"
34 PRINT:PRINT TAB(20)"3-Revision"
35 PRINT:PRINT TAB(20)"4-Fin de travail"
36 PRINT:PRINT:PRINT TAB(25)"Votre choix";
37 INPUT R$
38 R=VAL(R$)
39 IF ASC(R$)<49 AND ASC(R$)>53 THEN 36
40 ON R GOTO 45,200,300,400
44 REM
45 REM-ANGLAIS/FRANCAIS-
47 CLS
50 GOSUB 500
75 RJ=0:RF=0
80 FOR I=1 TO ND
90 C=0
100 PRINT M$(I,1)
120 INPUT R$
130 IF R$=M$(I,2) THEN 175
140 C=C+1:IF C>2 THEN 170
160 PRINT"RECOMMENCE":GOTO 100
170 PRINT "C'ETAIT: ";M$(I,2):RF=RF+1:GOTO 180
```

```
175 PRINT"BRAVO":RJ=RJ+1
180 NEXT I
190 GOSUB 550
195 GOTO 20
199 REM
200 REM FRANCAIS/ANGLAIS
201 REM------------------
205 CLS
210 GOSUB 500
220 RJ=0:RF=0
225 FOR I=1 TO ND
230 C=0
235 PRINT M$(I,2)
240 INPUT R$
250 IF R$=M$(I,1) THEN 290
255 C=C+1:IF C>2 THEN 265
260 PRINT"RECOMMENCE":GOTO 235
265 PRINT"C'ETAIT: ";M$(I,1):RF=RF+1:GOTO 180
290 PRINT"BRAVO":RJ=RJ+1
295 NEXT I
296 GOSUB 550
298 GOTO 20
299 REM
300 REM REVISION
301 REM------------------
305 CLS
310 GOSUB500
320 FOR I=1 TO ND
325 PRINT M$(I,1),M$(I,2)
330 NEXT I
335 INPUT"RETOUR AU MENU(O/N)";R$
340 IF R$="O"THEN 20 ELSE 400
399 REM
400 REM FIN DU TRAVAIL
401 REM------------------
405 GOTO 999
499 REM
500 REM SOUS-PROGRAMME DE REMPLISSAGE
501 REM------------------
505 READ ND
510 DIM M$(ND,2)
515 FOR I=1 TO ND
520 READ M$(I,1),M$(I,2)
525 NEXT I
530 RETURN
549 REM
550 REM BILAN
551 REM------------------
555 CLS:PRINT"REPONSE(S) JUSTE(S): "RJ
560 PRINT:PRINT"REPONSE(S) FAUSSE(S): "RF
565 INPUT"ON CONTINUE(O/N)";R$
570 IF R$="O" THEN 580 ELSE 400
580 GOTO 20
598 REM
599 REM DONNEES
600 REM------------------
605 DATA 10,TO GET,OBTENIR,TO PRINT,IMPRIMER,TO CLEAR,
    ECLAIRCIR
610 DATA THE END,LA FIN,NEW,NOUVEAU,TOLIST,LISTER,TO
    SAVE,SAUVEGARD
ER
615 DATA TO LOAD,CHARGER,LEFT,GAUCHE,RIGHT,DROITE
999 END
```

CHECK-LIST

DATA 167
READ 167
CLEAR 212
DIM (tableau de chaînes à une ou plusieurs dimensions) 212
ON... GOTO 194
FRE 213

TRAINING

- Amélioration des programmes proposés : fichier d'amis, calculs de date et de jours, programme d'enseignement.
- Tri de liste par la méthode proposée plus haut (programme + + + SEMAINE + + +) qui se nomme Tri Ripple.
- Création de grilles de mots croisés ou autres jeux sur les mots.

Enregistrer des données sur cassette

Stocker les données par le système des DATA nécessite la sauvegarde de l'ensemble du programme BASIC sur cassette. Opération que nous avons réalisée dès le programme STARTER 1, à l'aide de l'instruction CSAVE. Supposons, à présent, que nous voulions changer les données pour un même programme. Faudra-t-il, chaque fois, réécrire les DATA et sauvegarder l'ensemble du programme BASIC ? Nous risquerions d'être encombrés de programmes. C'est la raison pour laquelle ATMOS présente un système pratique de sauvegarde des données sur cassette ou disquette.

STORE et RECALL

Store et Recall sont les instructions qui permettent de réaliser les opérations d'enregistrement des données sur support magnétique (store) et de rappel de celles-ci en mémoire centrale (recall).
Ces données doivent se présenter obligatoirement sous forme de tableaux.

L'enregistrement

Il est réalisé grâce à STORE dont la syntaxe d'utilisation est :
STORE X, "Nom du tableau" (,S)
• X est l'identificateur du tableau déclaré au cours du programme par l'instruction DIM. Par exemple A$ si on a déclaré un tableau de chaînes de caractères à deux dimensions A$ (4,5)
• "nom du tableau" est le nom sous lequel vous décidez de l'enregistrer sur la bande magnétique ou la disquette. Ne pas omettre la virgule.
• (,S) est l'indice facultatif de vitesse lente de sauvegarde (300 bauds) ; par défaut, la vitesse sera normale (2400 bauds).
STORE A$, "BIDULE" (,S)
STORE T, "TEMPERATURE" (,S)
La marche à suivre est identique à celle de CSAVE. Le message SAVING suivi du nom du tableau apparaît en haut de l'écran. Il est accompagné d'une lettre spécifiant le type de données : R pour les nombres réels, I pour les entiers (Integer), S pour les chaînes de caractères (strings).

La lecture ou rappel des données

L'instruction RECALL permet de lire un tableau déjà enregistré sur cassette ou disquette. Le processus est identique à celui du chargement d'un programme BASIC par CLOAD. Auparavant, il est obligatoire d'avoir défini et déclaré dans le programme que ces données vont alimenter, un nouveau tableau de même type, de même dimension (ou plus grand), afin de recueillir les données en provenance de la cassette qui est rappelée en RAM ; ce sont les données qui sont lues dans un tableau et recopiées dans un autre.

La syntaxe de l'instruction est :
RECALL Y, "nom du tableau" (,S)

• Y est le nom du tableau où vont venir les données en provenance de la cassette.

Voici un premier exemple de *sauvegarde* de *données numériques,* à partir du programme TEMPÉRATURE :

```
5 'Sauvegarde d'un tableau
6 'de donnes numeriques
7 '-------------------------
10 DIM T(31)
15 FOR I= 1 TO 31
20 PRINT"JOUR NUMÈRO ";I;
30 INPUT T(I)
40 NEXT I
45 '-------------------------
50 PRINT"Appuyez sur une touche"
51 PRINT"des que le magnetophone"
52 PRINT"est pret"
60 GET A$
70 STORE T,"TEMPERATURE"
80 PRINT"Enregistrement termine"
85 '-----------------------------
90 END
```

Commentaires :

[10] Déclaration du tableau (31 jours)
[30] Entrée des températures du mois.
[50-60] Contrôle du déclenchement de l'enregistrement.
[70] Instruction d'enregistrement.
[80] Message de fin d'opération.

Voici, à présent, *la lecture* du même tableau :

```
5 'Lecture d'un tableau
6 'de donnes numeriques
7 '-------------------------
10 DIM T2(31)
50 PRINT"Appuyez sur une touche"
51 PRINT"des que le magnetophone"
52 PRINT"est pret"
60 GET A$
70 RECALL T2,"TEMPERATURE"
80 PRINT"Lecture terminee"
85 '-----------------------------
90 'Affichage
100 FOR I=1 TO 31
110 PRINT T2(I)
120 NEXT I
130 '-------------------------
140 END
```

Commentaires :

[10] Déclaration d'un tableau "destination" de même type et de dimension identique pouvant porter un nom différent du tableau "source".

[50-60] Contrôle et commande de déclenchement de la lecture.

[70] Instruction de rappel des données dans le tableau destination.

[80] Message de fin d'opération.

[90] Affichage du tableau "destination".

L'un des principaux intérêts du stockage des données est de pouvoir les transférer aisément d'un programme à un autre. A condition que les types de données et les dimensions des tableaux soient compatibles. Ceci peut être utile si votre CARNET D'ADRESSE augmente rapidement.

```
10 '
11 'CARNET D'ADRESSE 2
12 '*****************
13 '
20 CLS:PAPER7:INK0
30 PRINT@15,16;"Carnet d'adresses"
31 '
40 PRINT:PRINT TAB(15)"1-Remplir un carnet"
50 PRINT:PRINT TAB(15)"2-Lire un carnet"
60 PRINT:PRINT TAB(15)"3-Consulter un nom"
70 PRINT:PRINT TAB(15)"4-Fin de travail"
80 PRINT:PRINT:PRINT"Votre choix";
90 A$=KEY$:IF A$=""THEN 90 ELSE A=VAL(A$)
100 IF ASC(A$)<49 AND ASC(A$)>53 THEN 90
110 ON A GOTO 150,200,300,400
145 '
150 'creation du carnet
151 '
154 CLS:PAPER4:INK3
155 INPUT"Nombre de noms a enregistrer";N
160 DIM NOM$(N,4)
165 FOR I=1 TO N
170 PRINT"Numero";I;" (nom,prenom,adresse,tel"
180 INPUT"NOM:";N$:NOM$(I,1)=N$
185 INPUT"PRENOM";PNOM$:NOM$(I,2)=PNOM$
190 INPUT"ADRESSE ";AD$:NOM$(I,3)=AD$
192 INPUT"TEL:";TEL$:NOM$(I,4)=TEL$
193 NEXT I
194 PRINT"Voulez-vous enregistrer ce carnet(O/N)"
195 A$=KEY$:IF A$=""THEN 195
196 IF A$<>"O"AND A$<>"N"THEN 195
197 IF A$="O"THEN GOSUB 1000
198 GOTO 20
199 '
200 ' lecture carnet
205 '
210 CLS:PAPER2:INK0
220 PRINT"Ce carnet est-il   "
```

```
230 PRINT"1- Sur cassette?"
240 PRINT"2- En memoire vive?"
250 A$=KEY$:IF A$=""THEN250 ELSE A=VAL(A$)
260 ON A GOTO 2000,265
262 '
263 'affichage
264 '
265 FOR I=1 TO N
266 PRINTNOM$(I,1):PRINTNOM$(I,2)
267 PRINTNOM$(I,3):PRINTNOM$(I,4)
268 PRINT:PRINT
270 NEXT I
290 PRINT"Appuyez sur une touche "
291 PRINT"Pour revenir au menu."
295 GET A$:GOTO 20
299 '
300 'consultation
301 '
310 INPUT"Quel nom desirez-vous ";N$
320 FOR I=1 TO N
330 IF N$=NOM$(I,1)THEN GOSUB 600
335 NEXT I
350 PRINT"Voulez-vous continuer?"
360 GET A$:GOTO 20
599 '
600 'affichage Par nom
601 '
610 PRINTNOM$(I,1)+" ",NOM$(I,2)
620 PRINTNOM$(I,3)+" ",NOM$(I,4)
630 RETURN
999 '
1000 ' enregistrement
1001 '
1010 PRINT"Quand le magnetoPhone est Pret"
1020 PRINT"a enregistrer,"
1030 PRINT"appuyez sur une touche"
1035 PRINT"du clavier."
1040 GET A$
1050 STORE NOM$,"CARNET"
1060 PRINT:PRINT"Tableau enregistre"
1070 RETURN
1999 '
2000 'Lecture de tableau enregistre
2001 '
2005 CLEAR
2010 INPUT"Nombre de noms";N
2020 DIM NOM$(N,4)
2230 PRINT"Appuyez sur une touche"
2240 GET A$
2250 RECALL NOM$,"CARNET"
2260 GOTO 20
```

Commentaires :

30-110 Menu des procédures proposées.
30 : utilisations de PRINT pour centrer le titre au point de coordonnées colonne 15, ligne 6. Noter le point-virgule devant les données à afficher.

90-100 Emploi de KEY$ pour éviter d'avoir à valider chaque choix. ON...GOTO pour l'aiguillage des choix.

150-191 remplissage du carnet

194-197 appel du sous-programme de sauvegarde sur cassette.

200-260 lectures du carnet. Deux possibilités : le carnet peut être en mémoire vive car il vient d'être rempli ; le carnet doit être rappelé de la cassette en mémoire vive pour être lu.

265-270 affichage du carnet en mémoire vive.

300-370 consultation d'un nom sur le carnet en mémoire vive.

1000-1050 sous-programme d'enregistrement sur cassette.

2000-2050 sous programme de lecture d'un carnet déjà enregistré.
2005 : effacement de tout autre tableau en mémoire vive.
2020 : déclaration du tableau destination
2050 : rappel des données du source au destination
2260 : retour au menu pour lecture ou consultation

CHECK-LIST

STORE ...
RECALL ...
PRINT AT (PRINT)...
GET ...

TRAINING

A vous d'imaginer les améliorations du programme précédent ; suppression de noms dans un tableau, classement alphabétique, etc.

Dans la mémoire

Nous allons maintenant pousser ORIC dans ses derniers retranchements. C'est là qu'il va nous montrer toute sa puissance. Mais voyons, au préalable, comment est fait sa mémoire.

Structure de la mémoire

Vous savez que les ordinateurs ne comprennent que les nombres (les lettres sont traduites en nombres par le codes ASCII). Vous savez aussi que ces nombres sont gardés dans sa mémoire. Comment est faite cette mémoire ?
Ce sont de petits circuits électroniques miniaturisés (on dit intégrés).
Ces circuits comprennent des milliers de cases. Dans chaque case prend place un nombre.
Pour aller chercher un nombre l'ordinateur doit savoir où le trouver. Chaque case est repérée par un numéro. Les spécialistes l'appellent l'adresse.
Ainsi, ATMOS possède 65536 cases mémoire, numérotées de Ø à 65535.

Pour lire une case

Vous pouvez aisément lire le contenu d'une case. C'est le rôle de la fonction PEEK.
Essayez en mode immédiat :

```
?PEEK(54ØØØ)
145
Ready
```

La case mémoire d'adresse 54ØØØ contient le nombre 145.
Vous pouvez en tester d'autres.
Vous pouvez aussi les essayer systématiquement avec ce petit programme :

```
1Ø  CLS
2Ø  FOR I=65535 TO Ø STEP-1
3Ø  PRINT PEEK(I);
4Ø  NEXT I
RUN
```

Et vous voyez alors tout ce qu'ATMOS a dans la tête : une série de nombres. Remarquez qu'ils sont inférieurs à 255.
Laissez tourner le programme un certain temps, vous verrez alors que toutes les cases sont remplies avec le nombre 85. C'est pour cela que nous avons commencé par la fin (on dit les adresses hautes). Pour les cases de rang inférieur à 38912, le contenu est le même.
En réalité, on peut dire que les cases sont vides. Nous verrons plus loin pourquoi et comment on peut les remplir.

Contenu d'une case

Vous avez remarqué que tous les nombres lus par PEEK étaient inférieurs à 256. Vous savez certainement que les ordinateurs sont formés de circuits électriques ouverts ou fermés. En réalité, les circuits ne connaissent que deux nombres : Ø ou 1. On dit qu'ils comptent en base 2 ou en binaire.
Sans doute en est-il de même dans notre cerveau, mais depuis la nuit des temps, nous avons appris des choses et nous comptons maintenant en base 1Ø. (Nous avons 1Ø doigts).
Vous n'ignorez pas que lorsque vous écrivez un nombre, par exemple 6482 cela signifie en réalité 6ØØØ + 4ØØ + 8Ø + 2
ou $(6*10^3) + (4*1Ø^2) + (8*1Ø^1) + (2*1Ø^0)$.
Pour compter en binaire, c'est la même chose. Simplement, il n'y a que deux chiffres, Ø et 1, et les puissances de 1Ø sont remplacées par des puissances de 2.

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$

Ainsi 11Ø1 en binaire signifie $1*2^3 + 1*2^2 + Ø*2^1 + 1*2^0 = 8+4+Ø+1 = 13$ en décimal. Les cases mémoires d'ATMOS (et des microordinateurs) sont formées de 8 interrupteurs ouverts ou fermés correspondant chacun à un nombre binaire (on dit un bit).
Vous avez ainsi un nombre à 8 bits qu'on appelle un octet.
Ainsi la case Ø Ø Ø Ø 1 Ø Ø 1
contient le nombre ØØØØ1ØØ1.

Pour convertir en décimal, nous vous conseillons le truc suivant : écrivez au-dessus du nombre les puissances de 2 successives 128 64 32 16 8 4 2 1
Le nombre le plus grand que peut contenir une case est l'octet

1 1 1 1 1 1 1 1 soit $2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

Si vous faites le calcul, vous trouverez 255. Cela explique le résultat précédent. Vous voyez que les nombres binaires sont, pour nous, hermétiques et longs à manipuler.
Essayez, par exemple, d'additionner Ø1Ø1Ø111 avec ØØ111ØØ1 ?
Les gens qui manipulent les ordinateurs utilisent, le plus souvent, les nombres en base 16 (ou hexadécimaux).
Il faut alors 16 chiffres élémentaires. On utilise les chiffres habituels Ø...9 et 6 lettres A B C D E F.
Voici la correspondance :
décimal Ø 1 2 3 4 5 6 7 8 9 1Ø 11 12 13 14 15
hexa Ø 1 2 3 4 5 6 7 8 9 A B C D E F
Nous n'entrerons pas dans tous les détails. Sachez simplement qu'ATMOS sait compter en hexadécimal.

Essayez :
? # F
15
? # A
10
ou ? # FF
255
Il comprend les nombres hexadécimaux à condition de mettre un # devant.
Le dernier exemple vous montre que le plus grand nombre que peut contenir une case mémoire est FF, c'est-à-dire le plus grand nombre hexadécimal à deux chiffres.
Nous utiliserons peu l'hexadécimal. Sachez, cependant, qu'il est très employé par les informaticiens qui travaillent en langage machine (dernier chapitre). C'est simplement une représentation commode des octets et de leurs adresses.







Écrire en mémoire

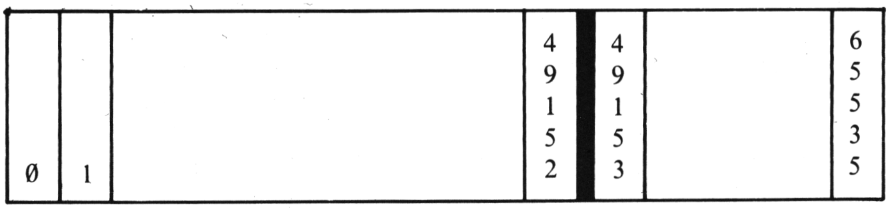
De plus en plus fort. Vous pouvez, vous-même et directement, modifier le contenu d'une case (ou d'autant de cases que vous voulez).
C'est le rôle de l'instruction POKE A,X
Attention POKE a deux arguments :
— A est l'adresse de la case: < = 65535
— X est le nombre que vous voulez y mettre: < = 255
C'est une instruction ; elle sera écrite directement dans une ligne.
Essayez POKE 2ØØØØ, 123
Il ne se passe rien d'autre.
Si vous regardez maintenant le contenu de la case 2ØØØØ par
?PEEK 2ØØØØ
123
Pour vérifier que cela modifie bien le contenu, essayez
?PEEK (23ØØØ). POKE 23ØØØ, 45: ?PEEK(23ØØØ)
RETURN
85
45
La case 23ØØØ qui contenait 85 (1er PEEK) contient maintenant 45 (2e PEEK).
Vous maîtrisez parfaitement ATMOS. Nous verrons par la suite tout ce que l'on peut faire avec ces nouvelles instructions.

R.A.M. et R.O.M.

Méfiez-vous cependant. Essayez, par exemple, la même chose que précédemment mais avec la case 53ØØØ. C'est facile avec l'éditeur :
?PEEK (53ØØØ): POKE 53ØØØ, 45:?PEEK (53ØØØ)
192
192

Ça ne fonctionne plus. Ne vous inquiétez pas. C'est normal. Il faut distinguer dans la mémoire deux parties fondamentalement différentes :
— La ROM (en anglais, READ ONLY MEMORY) : mémoire à lecture seule. Elle a été écrite une fois pour toute par le constructeur et vous ne pouvez plus la modifier. Ce sont tous les octets d'adresses comprises entre 49152 et 65535. C'est là que se trouve l'interpréteur BASIC qui permet à la machine de comprendre ce que vous lui écrivez.
— La RAM (RANDOM ACCES MEMORY) : mémoire à accès direct. Vous pouvez y lire et y écrire à volonté. C'est là que sont enregistrés vos programmes. Attention quand même, certaines zones sont utilisées par le BASIC pour y stocker des valeurs dont il a besoin. Si vous intervenez dans ces zones, vous risquez des perturbations dans le fonctionnement. Mais rassurez-vous : un RESET (sous le clavier) remettra tout en ordre.
La RAM et la ROM sont réalisées avec des circuits électroniques complètement différents. Une autre différence fondamentale entre RAM et ROM est la persistance de la mémoire. Si vous coupez l'alimentation électrique

d'ATMOS, la RAM "oublie" tous les nombres qu'elle conservait alors que la ROM les retient. C'est pourquoi vous êtes contraint d'enregistrer les programmes sur cassettes pour pouvoir les récupérer dans la RAM ultérieurement.



RAM
49152 octets 48 K

ROM
16384 octets

Topographie de la mémoire vive d'ORIC ATMOS

Adresses en mode texte

— de 49152 à 4912Ø : 32 octets inutilisés (zone tampon avec la ROM).
— de 4912Ø à 48ØØØ : 112Ø octets réservés à la gestion de l'écran texte.
— de 48ØØØ à 46Ø8Ø : 192Ø octets pour gérer le clavier alphanumérique et un deuxième clavier dit semi-graphique.
— de 46Ø8Ø à Ø : 46Ø8Ø octets pour les programmes Basic de l'utilisateur.

En mode graphique haute résolution

— de 49152 à 4912Ø : les 32 octets de séparation.
— de 4912Ø à 4Ø96Ø : 816Ø octets pour gérer l'écran haute résolution.
— de 4Ø96Ø à 39912 : 2Ø48 octets pour les 2 claviers
— de 38912 à Ø : 38912 octets pour les programmes en basic.

Graphique

Les caractères

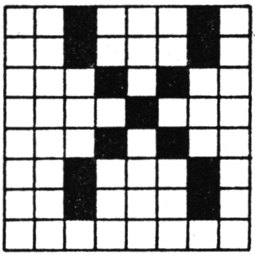
Comment sont dessinés les caractères

Tapez un X et regardez de près. Vous voyez nettement qu'il est formé de petits carrés placés les uns à côté des autres.
En réalité, tous ces caractères sont écrits dans une grille de 8 × 8 soit 64 petits carrés, (fig.). Les deux colonnes de gauche et la ligne du bas sont systématiquement vides pour permettre de bien séparer les caractères et les lignes.
Il faut qu'ATMOS puisse dessiner les caractères.

Le générateur de caractères

Lorsque vous appuyez sur la touche X, ATMOS sait que vous avez tapé sur la touche du code ASCII 88. Il a en sa possession uniquement ce numéro.
Il va alors chercher dans la mémoire ce qu'il faut deviner. Il le trouve, en mémoire, aux adresses 46784, 46785, ..., 46791. Soit 8 octets qui vont lui donner ce qu'il faut dessiner.
Comment cela se passe-t-il ?

46784
46785
46786
46787
46788
46789
46790
46791



Chaque ligne correspond à un octet. Sur une ligne, chaque carré éclairé correspond à un 1 binaire, chaque carré éteint à un zéro. Vous obtenez alors un nombre binaire dont la valeur est toujours inférieure à 64 puisque les deux bits de gauche sont toujours éteints.
Par exemple, essayez ?PEEK 46784
Vous obtenez : 34
En effet, regardez la première ligne. En mettant des zéros pour les carrés éteints et des uns pour les carrés allumés, vous obtenez l'octet suivant : ØØ1ØØØ1Ø soit $2^5 + 2^1 = 34$.
Vous avez compris. Il suffit de savoir convertir le binaire en décimal.
Vous trouverez par des PEEK successifs ou à partir du dessin :

46784	34
85	34
86	2Ø
87	16
88	2Ø
89	34
9Ø	34
91	Ø

Vous pouvez aussi vous amuser à regarder comment sont faits tous les caractères. Les huit octets correspondant à un caractère se trouvent aux adresses 46080 + A*8 à 46080 + A*8 + 7, A étant le code ASCII du caractère. Par exemple, pour un X, vous retrouvez 46080 + 88*8 = 46784. Pour le A, ce sera 46600 à 46607.

Modification des caractères

Si vous avez bien compris la différence entre ROM et RAM, vous comprendrez aisément ce qui va suivre.
Dans un ordinateur, les caractères sont stockés en ROM. Il faut que la machine se souvienne, dès la mise en route, de ses lettres pour pouvoir les faire apparaître sur l'écran. C'est le cas d'ATMOS. Mais ATMOS se distingue de beaucoup d'autres, par le fait que, à la mise en route ou lors d'un "reset", il recopie tous les caractères en RAM dans la zone de mémoire d'adresses comprises entre 46080 et 48000. Et ce sont ces caractères-là qui sont utilisés par ATMOS.

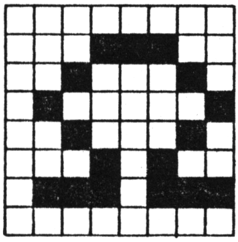
Nous avons vu précédemment que la ROM était aux adresses supérieures à 49152. Nous sommes donc bien en RAM. L'avantage ? Contrairement aux autres ordinateurs, vous pouvez modifier tous les caractères comme vous le voulez.
Voici un exemple simple. Essayez ce programme. Il vous permettra de faire une farce à un ami, possesseur d'ATMOS :

```
20  FOR I=46080 TO 47104
30  POKE I,0
40  NEXT I
50  NEW
RUN
```

Vous voyez les caractères disparaître les uns après les autres. Tapez sur le clavier : vous n'avez plus de caractère sur l'écran. ATMOS a perdu ses lettres ! Un conseil : vendez la mèche à votre ami, avant qu'il ne téléphone au réparateur. Et un RESET remettra tout en ordre.
Vous avez immédiatement compris que ce programme mettait à zéro tous les octets des lignes de tous les caractères. Et il n'y a aucun point allumé. Vous pouvez de la même façon permuter toutes les lettres. Cela constituera un bon exercice sur les boucles, les PEEK et les POKE.

```
20  FOR A=90 TO 64 STEP-1
30  FOR N=0 TO 7
40  X=PEEK (46080+A*8+N)
50  POKE 46088+A*8+N,X
60  NEXT N
70  NEXT A
```

Expliquons-nous : pour une valeur de A fixée dans la première boucle (par exemple A = 8Ø, lettre P), regardons ce que fait la deuxième boucle : elle prend un par un les octets qui définissent la lettre P (ligne 4Ø) et elle le transfère dans les cases qui définissent la lettre Q (ligne 5Ø). Ainsi le Q devient P.
Grâce à la première boucle, le programme fait cela pour tous les caractères en commençant, par Z. Ainsi Z devient Y, Y devient X, X devient W etc. Un moyen comme un autre de rendre vos listings incompréhensibles.
Vous pouvez ainsi créer vos propres caractères. Si vous souhaitez remplacer le caractère O par le caractère grec Ω. Faites un dessin. Numérotez les colonnes avec les puissances de 2 successives afin de faciliter le calcul.



Calculez chaque octet.
Le programme suivant vous permettra alors d'entrer vos nombres dans les octets.

```
 2Ø PRINT "TAPEZ LE CARACTERE QUE VOUS VOULEZ
    MODIFIER
 3Ø GET A$ :A=ASC(A$)
 4Ø PRINT "DONNEZ LES 8 NOMBRES"
 5Ø FOR I=Ø TO 7
 6Ø PRINT "LIGNE NUMERO";I+1
 7Ø INPUT X: POKE 46Ø8Ø+A*8+I,X
 8Ø NEXT I
 9Ø PRINT "VOICI LE NOUVEAU CARACTERE"; CHR$(A)
1ØØ GOTO 2Ø
```

Vous pouvez aussi :
— remplacer les caractères par des dessins. Essayez, par exemple, avec les huit nombres suivants : Ø, 36, 24, 6Ø, 126, 129, Ø, Ø ;
— modifier le jeu de caractères graphiques.
Vous aurez, ainsi, un fonctionnement normal de la machine en mode TEXT et en mode LORES Ø ainsi que les caractères graphiques de votre choix en LORES 1.

Programme pour myopes

(Un exemple de programme amusant) :

```
10 REM+++LETTRES+++
15 CLS:PAPER2:INK5
18 PRINT"TAPEZ UNE LETTRE"
20 A$=KEY$:IF A$="" THEN20 ELSE A=ASC(A$)
23 REM
25 REM ON VA CHEORCHER LES OCTETS
30 FOR N=0 TO 7
40 Y(N)=PEEK(46080+8*A+N)
50 NEXT N
98 REM
99 REM AFFICHAGE
100 FOR N=0 TO 7
110 Y=Y(N):GOSUB 1000
120 FOR M=1 TO 8
125 IFX(M)=1 THEN PRINTA$;
126 IF X(M)=0 THEN PRINT" ";
127 NEXT M
130 PRINT
140 NEXT N
200 GOTO 18
999 REM
1000 REM CONVERSION EN BINAIRE
1010 C=128: FOR M=1 TO 8
1020 B=Y-C:IF B<0 THEN X(M)=0:GOTO 1040
1030 X(M)=1:Y=B
1040 C=C/2:NEXT M
1050 RETURN
```

Vous verrez les lettres s'afficher en gros, chaque point de la lettre étant une petite lettre.
Ce programme est intéressant car il utilise un sous-programme dont vous pourrez vous servir très souvent.
Le sous-programme 1ØØØ à 1Ø5Ø convertit un nombre décimal en binaire. Plus précisément, si vous lui envoyez un nombre Y< =256), il vous renvoie dans un tableau X(M) les valeurs Ø ou 1 correspondant aux huit bits du nombre binaire.
Le sous-programme fonctionne de la façon suivante : on retranche au nombre Y un nombre C qui vaudra successivement 128, 64, 32... (ligne 1Ø2).
Si la différence est < Ø, le premier bit est nul. Alors X(M) vaut Ø.
On recommence avec C/2 (ligne 1Ø4Ø). Si la différence est ≥Ø alors X(M) vaut 1.
Et on recommence avec C/2 et la différence calculée précédemment (ligne 1Ø3Ø). Le reste du programme (lignes 99 à 14Ø) consiste à afficher le résultat pour chaque octet.

Un programme pour ceux qui louchent

Vous garderez une grande partie du programme précédent. Celui-ci utilise le même sous-programme.
Nous vous laissons le soin de comprendre le reste listez le programme, lancez-le puis observez le changement de lignes. Et si cela vous amuse, vous pouvez compléter le programme pour qu'il écrive de droite à gauche.
Bonne chance. Si vous vous faites du souci pour la santé des auteurs, faites tournez le programme, soyez patient...
Quand il aura terminé son travail, regardez votre programme dans un miroir...

```
10 REM+++RIORIM EMMARGORP+++
20 FOR A=48 TO 90
23 REM
25 REM ON VA CHERCHER LES OCTETS
30 FOR N=0 TO 7
40 Y(N)=PEEK(46080+8*A+N)
50 NEXT N
98 REM
100 FOR N=0 TO 7
110 Y=Y(N):GOSUB 1000
130 GOSUB 3000 :POKE 46080+8*A+N,Y
140 NEXT N
200 NEXT A
998 END
999 REM
1000 REM CONVERSION EN BINAIRE
1010 C=128: FOR M=1 TO 8
1020 B=Y-C:IF B<0 THEN X(M)=0:GOTO 1040
1030 X(M)=1:Y=B
1040 C=C/2:NEXT M
1050 RETURN
2000 REM INVERSION
2010 FOR M=1 TO 4
2020 AUX=X(M):X(M)=X(9-M):X(9-M)=AUX
2030 NEXT M
2040 RETURN
3000 REM CONVERSION DECIMALE
3010 Y=0
3020 FOR M=1 TO 8
3030 Y=Y+X(M)*2^(M-2):NEXT M
3040 RETURN
```

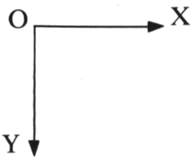
N'oubliez pas CTRL C si vous voulez interrompre ce cauchemar avant la fin.
Un coup de Reset remettra tout en ordre quand vous le voudrez.

Graphiques haute résolution

Nous avons déjà vu les graphiques basse résolution. Ils permettent de faire des dessins jolis mais assez grossiers. Nous allons maintenant aborder des dessins plus fins.

Caractéristiques de l'écran

Tapez au clavier HIRES : l'écran devient noir et si vous tapez d'autres lettres, elles ne s'affichent que sur les trois lignes du bas. HIRES signifie en anglais High Resolution. La partie du cadran en noir est destinée à recevoir des graphiques (points, lignes, cercles, figures géométriques).
Pour sortir du mode HIRES, tapez TEXT, LORES Ø ou LORES 1 suivant ce que vous désirez faire.
Pour effacer l'écran graphique, utilisez HIRES mais votre dessin est perdu.
En mode haute résolution, vous avez directement accès aux points qui constituaient les lettres. Pour cela, il faut les repérer sur l'écran. On utilise comme en mathématique deux axes avec les différences qui suivent :
— les axes ne sont pas visibles
— l'axe des Y est orienté vers le bas.
L'axe horizontal est gradué de Ø à 239 ;
L'axe vertical de Ø à 199.
L'écran est ainsi divisé en 24Ø*2ØØ soit 48ØØØ points.
C'est beaucoup et c'est pourquoi on appelle ce fonctionnement "Haute Résolution".



Le curseur graphique

Rappelez-vous le curseur d'écriture du mode TEXT. Un rectangle clignotant qui se déplace lorsque l'on écrit.
Ici, vous ne voyez rien. C'est un peu plus compliqué. En réalité le curseur existe, il a au départ comme coordonnées Ø,Ø.
Vous pouvez le déplacer et le faire apparaître ou disparaître avec l'instruction CURSET. Cette instruction est de la forme :
CURSET X,Y,P
X et Y sont l'abscisse et l'ordonnée de la nouvelle fonction. P est un paramètre qui permet d'allumer ou d'éteindre le curseur.
Essayons

```
1Ø  INPUT P
2Ø  CURSET 1ØØ, 1ØØ,P
3Ø  GOTO 1Ø
RUN
    ?1
```

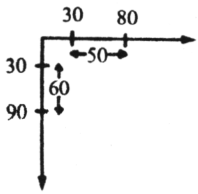
Un petit point apparaît au milieu de l'écran. Ø le point disparaît. 1 le point réapparaît.

Essayons maintenant 2. Le point disparaît. Encore 2. Le point réapparaît. 3. Le point reste. Encore 3. Le point reste. Ø il disparaît.
Ø. fait un point de la couleur du papier
1. fait un point de la couleur de l'encre
2. fait un point de la couleur opposée.
C'est-à-dire fait apparaître un point, s'il n'y a rien ; et le fait disparaître, s'il est là.
3. ne fait rien. Se contente de mettre le curseur dans la position indiquée.
On peut tout faire avec CURSET.

Par exemple :

```
1Ø  HIRES
2Ø  FOR J=1 TO 15Ø
3Ø  CURSET J,J,1
4Ø  NEXT J
```

Trace un trait oblique.
Toutes les figures géométriques sont représentables, à condition de connaître les équations correspondantes.
La couleur du papier et celle de l'encre peuvent varier. Seule les 3 lignes du bas restent noir et blanc. Un nouvel HIRES ramènera toujours la partie graphique en noir et blanc.
Si vous savez où se trouve le curseur, vous aurez souvent intérêt à lui faire effectuer un déplacement relatif.
Le curseur étant en 3Ø, 3Ø un déplacement relatif augmentera ses coordonnées. Vous utiliserez alors CURMOV X,Y,P. Par exemple, dans le cas précédent, CURMOV 5Ø, 6Ø, 1 amènera le curseur en 8Ø, 9Ø et allumera le point correspond. Vous pouvez de la même façon que précédemment tracer un trait.



Tracer des traits
Pour tracer un trait, il y a beaucoup mieux : c'est l'instruction DRAW
DRAW X,Y,P
trace un segment de droite entre la position actuelle du curseur et la position dont les coordonnées ont été augmentées de X et Y. P joue le même rôle que précédemment.
Essayons :

```
1Ø  HIRES
2Ø  CURSET 1,1,1
3Ø  DRAW 149,149,1
```

Nous obtenons le même trait que précédemment.
Pour tracer un rectangle, c'est très simple

```
10  HIRES : PAPER 5: INK 3
20  CURSET 30,30,1
30  DRAW 50,0,1
40  DRAW 0,60,1
50  DRAW  -50,0,1
60  DRAW 0,-60,1
```

Voilà un rectangle. Pour bien comprendre l'intérêt de la commande DRAW, modifions le programme de la façon suivante.

```
15  FOR X=10 TO 140 STEP 12
20  CURSET X,X,1
.
.
.

70  NEXT X
```

Le même rectangle est reproduit plusieurs fois mais translaté.

Faire des ronds

Faire des traits n'est pas à la portée de n'importe quel ordinateur. Mais avec les cercles, nous rejoignons le gratin de la micro-informatique. Et c'est très facile.
Il faut d'abord positionner le centre par CURSET. Ensuite, le cercle se fait automatiquement par CIRCLE R,P
R est le rayon et P le même paramètre que CURSET et DRAW. Le cercle est centré sur le point défini par CURSET.

```
10  HIRES
20  CURSET 120, 100, 0
30  CIRCLE 100, 1
```

Dans cette instruction comme dans les précédentes, le curseur ne doit jamais sortir de l'écran 0<=X=239; 0<=Y<=199 : il faut bien calculer votre coup.
En jouant sur CURSET et sur R, vous pouvez faire de beaux dessins. Par exemple :

```
 5  REM+++ RONDS+++
10  HIRES
15  PAPER 4: INK 3
16  WAIT 50
```

```
17  FOR X=1 TO 91 STEP 1Ø
2Ø  CURSET X+12, 199-X,3
3Ø  CIRCLE X,1
4Ø  NEXT X
```

Et si vous voulez maintenant un ballon qui se gonfle

```
 4  REM BALLON
 5  REM
1Ø  HIRES
2Ø  PAPER Ø: INK 3
3Ø  CURSET 12Ø, 1ØØ, 1
4Ø  FOR R=1 TO 9Ø
5Ø  CIRCLE R,1
6Ø  NEXT R
```

qui se dégonfle

```
 7Ø  WAIT 2ØØ
 8Ø  FOR R=98 TO 1 STEP -1
 9Ø  CIRCLE R,Ø
1ØØ  NEXT R
```

ou qui explose

```
7Ø  HIRES: EXPLODE: WAIT 3ØØ
```

Mais il peut changer de couleur en rajoutant :
45 INK R/15+1

Traits pleins ou pointillés

Nous savons faire des figures géométriques. Mais il est parfois utile de distinguer plusieurs figures. C'est possible soit par des pointillés de différentes tailles, soit par des couleurs différentes.

— Pointillés
Grâce à l'instruction PATTERN, nous pouvons avoir des traits ou des ronds en pointillés.
Par exemple :

```
1Ø  HIRES
2Ø  PAPER 4: INK 1: CURSET 1Ø, 1Ø, 1
3Ø  DRAW 1ØØ,1Ø,1
4Ø  PATTERN 17Ø
5Ø  CURSET 1Ø,2Ø,1
6Ø  DRAW 1ØØ,Ø,1
```

Le trait du dessous est pointillé. Comment fonctionne ce PATTERN ? ATMOS travaille avec des nombres de 8 chiffres binaires. (Voir 1re partie). Ce sont des octets. Pour tracer des traits, il travaille également sur des octets. PATTERN lui indique combien de points il doit allumer sur 8 points. Comment cela ? Chaque bit de l'octet correspond à un point. Si le bit est à 1 le point est allumé, s'il est à Ø, il est éteint.
Par exemple 1 1 1 1 1 1 1 1
donnera : ____
Trait continu.
Attention, la figure est très élargie. 8 points représentent environ 1 cm. Nous aurons les pointillés que nous voudrons en composant l'octet correspondant. Et nous composons l'octet par sa valeur en décimal dans PATTERN.
Dans l'exemple précédent, PATTERN 17Ø correspondait à :
1 Ø 1 Ø 1 Ø 1 Ø (128 + 32 + 8 + 2)
d'où 1 point allumé sur deux, soit ....
Nous aurions pu prendre 15 soit : --
Ø Ø Ø Ø 1 1 1 1
qui donne de gros pointillés : —
Ce programme nous permettra d'essayer toutes les combinaisons :

```
1Ø  HIRES
2Ø  PAPER 4: INK 1
25  FOR Y=1 TO 198 STEP 4
3Ø  INPUT "PATTERN";X
4Ø  PATTERN X
5Ø  CURSET 1Ø,Y,1
6Ø  DRAW 19Ø,Ø,1
7Ø  NEXT
```

Certains traits se ressemblent. Les plus intéressants semblent être pour 1,15,122 ou 254. Choisissez.
Nous pouvons aussi essayer avec des cercles en rajoutant :
35 HIRES : PAPER 4 : INK 1
et en substituant à 5Ø et 6Ø

```
5Ø  CURSET 12Ø, 1ØØ, Ø
6Ø  CIRCLE 9Ø,1
```

Couleurs

La couleur du fond ou de l'encre peut être imposée dans des zones localisées de l'écran grâce à la commande FILL.
FILL ressemble un peu à PLOT du mode basse résolution, lorsqu'il était utilisé avec des attributs (ou caractères préfixés). Rappelons-nous, c'était ainsi que l'on imposait la couleur des caractères ou du fond. Souvenons-nous aussi que là où nous imposions un attribut, nous ne pouvions pas mettre de caractères sans supprimer l'attribut.

• *Structure de l'écran*

Revenons un peu sur la mémoire. En mode haute résolution, l'écran comporte 240*200 points. En réalité, les points sont regroupés horizontalement par 6. Nous aurons donc sur une ligne 40 groupements de 6 points. Appelons-les des segments.

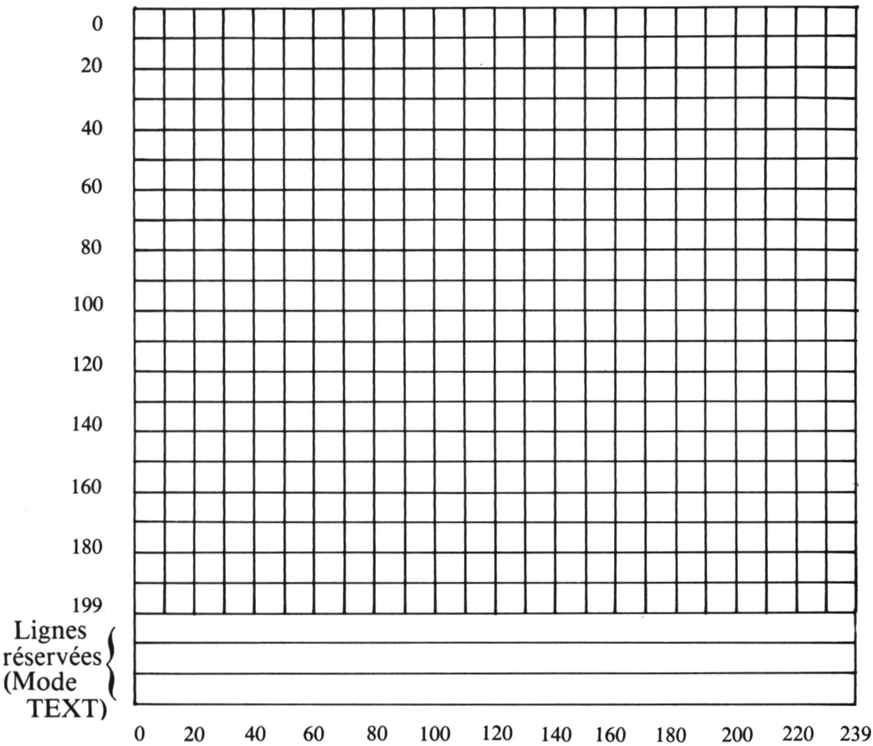
Et nous aurons 200 lignes identiques à celle-ci. Ce qui donne finalement 40 colonnes verticales. Chaque colonne est formée de 200 segments. A chaque segment est associé un octet en mémoire.

Il y a 40*200 = 8000 segments donc 8000 octets en mémoire réservés à l'écran en haute résolution.

Vous pouvez accéder directement aux octets de la mémoire d'écran par des POKE.

Essayez par exemple : POKE 42020,65

Vous verrez un point s'allumer. Tout dépend de ce que vous mettez dans l'octet correspondant.



Mode haute résolution (HIRES)

• *Paramètres utilisés dans FILL*

FILL remplit un certain nombre de segments, c'est-à-dire d'octets de la mémoire d'écran. Cette instruction comporte 3 paramètres : FILL A,B,C.

Attention, FILL agit à partir de la position courante du curseur et elle déplace le curseur verticalement mais pas horizontalement.

Exemple CURSET 15,1Ø,1
FILL 2Ø, 3, C
Le curseur est dans la 3e colonne. Le 2e paramètre indique le nombre de colonnes remplies, ici 3 : les colonnes 3,4,5 seront remplies.

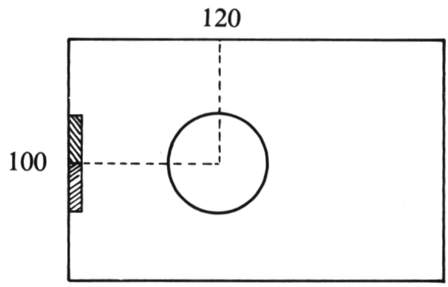
120 100

Le 1er paramètre indique le nombre de lignes. Ici 2Ø lignes seront affectées par FILL.
Résumons :
1er paramètre. Nombre de lignes < à 2ØØ si le curseur est au Ø, <2ØØ – Y si le curseur est sur la ligne Y.
2e paramètre. Nombre de colonnes < 4Ø si le curseur est au Ø, <40 – N si N est le numéro de la colonne où se trouve le curseur.

• *Troisième paramètre*
C'est lui qui commande toute l'action de FILL.
— Si C est compris entre Ø et 31, il se comporte exactement comme un attribut de l'instruction PLOT.
Rappelons :
de Ø à 7 couleur de l'encre
de 8 à 15 clignotement
de 16 à 23 couleur du papier
de 24 à 31 à éviter.
Rappelons que dans ce cas, l'attribut affecte toute la ligne et qu'il efface ce qui était sur l'écran à cet endroit.

Par exemple :

```
1Ø  HIRES
2Ø  CURSET 12Ø,1ØØ,1
3Ø  CIRCLE 3Ø,1
4Ø  WAIT 1ØØ
5Ø  CURSET Ø,7Ø,1
6Ø  FILL 3Ø,1,2
7Ø  WAIT 1ØØ
8Ø  FILL 3Ø,1,19
```



La première instruction FILL ne fait rien là où on l'applique. (Rectangle hachuré). Mais elle colore tout ce qui est à droite en vert. Ici le demi-cercle du haut.
La deuxième colore tout le fond à droite en jaune.

— Si C est compris entre 32 et 125, l'instruction remplit effectivement le rectangle correspondant avec un motif qui dépend de la valeur. La valeur la plus simple est 63 qui remplit complètement le rectangle.
Certaines valeurs donnent les mêmes résultats. Par exemple :
63, 95 et 127. Mais avec des nuances. Si vous écrivez par-dessus ou modifiez

l'encre ou le papier, vous observerez des différences. L'étude détaillée nous entraînerait un peu loin. Entraînez-vous.
Voici un petit programme qui ne sert à rien. Est-ce l'ATMOS dans l'hyper-espace ?

```
90 REM+++RIEN+++
100 HIRES:CURSET 120,100,0
110 CIRCLE 90,1
114 CURSET 60,50,1
115 FILL 100,20,88
120 CURSET 0,0,0
130 FOR Y=1 TO 99 STEP 1
140 FILL 2,1,Y-7*INT(Y/7)+1
150 NEXTY
160 CURSET 0,0,0:DRAW 239,199,1
170 CURSET 120,100,0:CIRCLE 50,1
180 CURSET 0,199,0:DRAW 239,-199,1
10 WAIT 200
200 FOR I=0 TO7:INK I:PAPER 7-I
210 WAIT 100:NEXTI
220 GOTO114
```

Des courbes et du texte

La haute résolution est parfaite pour l'étude des courbes ou des résultats d'expériences physiques. Pas de problème particulier. Voici quelques exemples :

```
10 REM+++SINUSOIDES+++
20 HIRES
30 PAPER3:INK4
35 Z1=10
40 FOR X=0 TO 239
50 Y=100-90*SIN(8*PI*X/239)
55 Z=100-90*COS(8*PI*X/239)
60 CURSET X,Y,1
65 DRAW X1-X,Y1-Y,1
66 CURSET X,Z,1
67 X1=X:Y1=Y
68 Z1=Z
70 NEXT X
```

Remarquez simplement la ligne 5Ø. Le 1ØØ – 9Ø ... est dû au fait que l'axe des Y est inversé et que le zéro est en haut. Vous êtes peut-être déçu de ne pas obtenir une courbe continue. C'est possible : il suffit de tracer un petit segment entre chaque point.

Le problème est qu'il faut connaître les coordonnées du point précédent. Nous allons mémoriser à la fin de la boucle X et Y dans deux autres variables X1 et Y1 afin de pouvoir les utiliser dans la bouche suivante.
d'ou 67 X1 = X : Y1 = Y
Il suffit alors de tracer avant ceci les segments entre les points par la ligne 65 DRAW X1 – X, Y1 – Y,1
Remarquons que ces valeurs de X1 et Y1 devraient ne pas être définies pour la première boucle. Il faudrait donc rajouter une ligne avant la boucle pour les initialiser.
Par exemple :
35 X1 = Ø; Y1 = Ø
Mais souvenez-vous que avec RUN, toutes les variables sont remises à Ø et cette ligne est donc facultative.
Nous pouvons sur le même graphique ajouter la fonction COS.
Il nous suffira d'ajouter
55 Z = 1ØØ – 9Ø + COS (8*PI*(X – 12)/239)
et 66 CURSET X,Z,1
La fonction COS apparaîtra alors en pointillé. Si nous voulons une courbe amortie, remplaçons la ligne 55 par
55 Z = 1ØØ – 9Ø*EXP (– (X – 121ØØ)*COS 68*PI*(X – 12)/239)
Enfin, pour faire apparaître des axes et dire ce que l'on fait. Un axe horizontal rouge apparaîtra avec
32 CURSET 13,1ØØ,1: FILL 1,1,17
Un axe vertical par
31 CURSET 12,Ø,1 : DRAW Ø,199,1
Enfin, pour faire apparaître du texte, deux solutions :
Nous disposons de 3 lignes de texte en bas de l'écran.
On peut y écrire par l'instruction PRINT habituelle. Par exemple ici :
25 PRINT "TRAITS PLEINS: SINUS"
26 PRINT "POINTILLÉS : COSINUS AMORTI";
N'oublions pas alors d'ajouter en fin de programme une boucle 8Ø GOTO 8Ø

pour éviter de perdre notre texte à la fin.
Mais ATMOS nous autorise également à mettre du texte dans le graphique lui-même par l'instruction CHAR.
Avant d'utiliser CHAR, il faut positionner le curseur graphique pour lui dire où nous voulons placer le caractère. Cela se fait bien sûr sur CURSET.
Ensuite CHAR A,B,P placera le caractère de code ASCII A à cet endroit. B indique s'il s'agit d'un caractère normal (o) ou semi-graphique 1. P est le paramètre habituel.
Dans ce programme, nous pouvons mettre un X à l'extrémité de l'axe des X par
33 CURSET 23Ø,9Ø,Ø: CHAR 88,Ø,1
Il est possible ainsi d'écrire des mots ou des phrases. C'est un peu plus compliqué : il faut faire une boucle qui lit les caractères les uns après les autres (voir chapitre sur les chaînes de caractères : fonction MID$)
On peut ainsi les placer les uns après les autres en les décalant.

Voici le programme complet :

```
10 REM+++COURBES+++
20 HIRES
25 PRINT"TRAITS PLEINS:SINUS
26 PRINT"POINTILLES    :COSINUS AMORTI";
30 PAPER3:INK4
31 CURSET 12,0,1:DRAW 0,199,1
32 CURSET 13,100,1:FILL 1,1,17
33 CURSET 230,90,0:CHAR 88,0,1
35 Z1=10
40 FOR X=12TO 239
50 Y=100-90*SIN(8*PI*(X-12)/239)
55 Z=100-90*EXP(-(X-12)/100)*COS(8*PI*(X-12)/239)
60 CURSET X,Y,1
65 DRAW X1-X,Y1-Y,1
66 CURSET X,Z,1
67 X1=X:Y1=Y
68 Z1=Z
70 NEXT X
80 GOTO 80
```

Voyage dans l'espace

Et voici pour finir, le plus étonnant. ATMOS sort de son écran. Voici la télévision en relief. Nous avons la représentation d'un volume dans l'espace. Il tourne autour de trois axes X Y Z en appuyant sur la touche correspondante. Le programme fait appel à quelques notions mathématiques pour les rotations (revoyez vos manuels de géométrie dans l'espace !)
Le reste n'est qu'une application de notions déjà vues.
ET BON VOYAGE...

```
5 REM+++CUBE+++
9 TEXT
10 HIRES:DIM X(10):DIM Y(10):DIM Z(10)
20 DATA 50,-50,50,50,50,50
21 DATA -50,50,50,-50,-50,50,50,-50,50
30 DATA 50,-50,-50,50,50,-50
31 DATA -50,50,-50,-50,-50,-50,50,-50,-50
50 FOR N=1 TO 10:READ X(N),Y(N),Z(N):NEXT N
60 CO=.995:SI=.0998
70 P=1
80 INK5:GOSUB 1000
90 PRINT"APPUYEZ SUR X,Y OU Z"
10 A$=KEY$:IF A$="" THEN 100
102 PRINT:PRINT"APPUYEZ SUR X,Y OU Z"
105 P=0:GOSUB 1000
110 IF A$="Z" THEN GOSUB 2000
120 IF A$="Y" THEN GOSUB 3000
130 IF A$="X" THEN GOSUB 4000
```

```
140 P=1:GOSUB 1000
150 GOTO 100
999 REM
1000 REM Trace du cube
1010 FOR N=1 TO 4
1020 CURSET X(N)+100,Y(N)+100,P
1030 DRAW X(N+1)-X(N),Y(N+1)-Y(N),P
1040 NEXT N
1110 FOR N=6 TO 9
1120 CURSET X(N)+100,Y(N)+100,P
1130 DRAW X(N+1)-X(N),Y(N+1)-Y(N),P
1140 NEXT N
1210 FOR N=1 TO 4
1220 CURSET X(N)+100,Y(N)+100,P
1225 IF X(N)=X(N+5) AND Y(N)=Y(N+5)THEN 1240
1230 DRAW X(N+5)-X(N),Y(N+5)-Y(N),P
1240 NEXT N
1300 RETURN
1999 REM
2000 REM Rotation autour de Z
2010 FOR N=1 TO 10
2020 X(N)=X(N)*CO+Y(N)*SI
2021 Y(N)=-X(N)*SI+Y(N)*CO:NEXT N:RETURN
2999 REM
3000 REM Roation autour de Y
3010 FOR N=1 TO 10
3020 X(N)=X(N)*CO+Z(N)*SI
3021 Z(N)=-X(N)*SI+Z(N)*CO:NEXT N:RETURN
3999 REM
4000 REM Rotation autour de X
4010 FO N=1 TO 10
4020 Y(N)=Y(N)*CO+Z(N)*SI
4021 Z(N)=-Y(N)*SI+Z(N)*CO:NEXT N:RETURN
```

CHECK-LIST

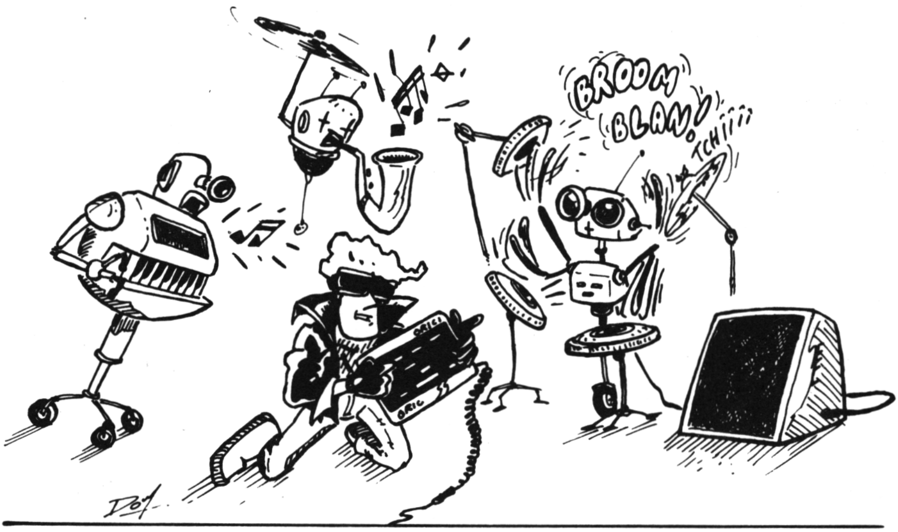
PEEK 214
POKE 214
conversion hexadécimal/décimal
Notions sur ROM et RAM
Générateur de caractères
HIRES 200
CURSET 202
CURMOV 203
DRAW 205
PATTERN 205
FILL 206
CHAR 206

TRAINING

• Programme qui écrit de droite à gauche (à la suite de celui qui inverse les lettres).
• Programme qui retourne les lettres la tête en bas.
• Programme qui écrit en très grosses lettres (réalisées avec des carrés colorés : mode LORES Ø).
• Améliorez le programme du cube en traçant en pointillé la partie cachée (difficile).

Niveaux, fréquences et enveloppes

Nous avons vu dans le chapitre 2 comment faire du bruit, des sons ou de la musique.
Toutes les instructions concernant les possibilités sonores d'ATMOS ont été alors étudiées. Mais vos connaissances en programmation n'étaient pas alors suffisantes pour exploiter à fond toutes les possibilités de votre micro.
Le moment est venu de faire la synthèse. Nous vous proposons deux programmes qui abordent tous les domaines de la programmation. Le premier vous permettra de jouer du piano ou d'autres instruments sur ATMOS. L'autre vous offrira la possibilité de tester votre oreille ou celle de vos amis ou enfants.



Jouez du synthétiseur

Nous avons déjà réalisé une petite orgue rudimentaire. Mais ses possibilités étaient limitées. Les notes étaient peu nombreuses et on ne distinguait pas les tons des demi-tons.
Ici, nous allons plus loin :
— la première rangée du clavier (de Z à M) joue les notes ;
— la deuxième (S,D,G,H...) joue les dièzes ;
— les flêches [←] et [→] vous permettent de changer d'octave. Vous pouvez ainsi couvrir de façon continue six octaves.
— la touche [RETURN] vous offre le choix entre deux enveloppes : piano ou orgue.
— les flèches [↑] et [↓] augmentent ou diminuent la durée des notes.
— Pour arrêter, touche [ESC]
Amusez-vous bien !

```
5 REM+++SYNTHETISEUR+++
20 PRINTCHR$(6)
30 DIMNO$(100)
40 DIM NO(100)
45 FOR N=1 TO 100:NO(N)=-1:NO$(N)="COUAC"
46 NEXT N
50 DATA 90,1,DO,83,2,DO#,88,3,RE,68,4,RE#
55 DATA 67,5,MI,86,6,FA
60 DATA 71,7,FA#,66,8,SOL,72,9,SOL#,78

65 DATA 10,LA,74,11,LA#,77,12,SI
70 FOR I=1 TO 12
80 READ A:READ NO(A):READ NO$(A)
90 NEXT I
100 O=3:C=1:E=1:D=500
110 CLS:PAPER 4:INK 1
200 A$=KEY$:IF A$="" THEN 200 ELSE A=ASC(A$)
220 IF A<8 THEN 200:IF A>13 AND A<48 THEN 200
230 IF A>55 AND A<66THEN 200:I A>90 THEN 200
240 IF A=12 THEN END
300 IF A>65 AND A<91 THEN GOTO 1000
400 IF A=8 THEN 410 ELSE 500
410 IF O=0 THEN 200 ELSE O=O-1
500 IF A=09 THEN 510 ELSE 600
510 IF O=6 THEN 200 ELSE O=O+1
600 IF A=13 THEN 610 ELSE 700
610 IF E=1 THEN E=2 ELSE E=1
700 IF A=11 THEN 710 ELSE 800
710 IF D=32700 THEN 200 ELSE D=80+D
800 IF A= 10 THEN 810 ELSE 900
810 IF D=80 THEN 200 ELSE D=D-80
900 IF A>=48 AND.A<=55 THEN 910 ELSE 980
910 IF A=48 THEN 200 ELSE C=VAL(A$)
980 GOTO 200
1000 IF NO(A)<0 THEN 1001 ELSE 1005
1001 PING:WAIT 50:PRINT NO$(A):GOTO 200
1005 MUSIC 1,O,NO(A),0
1010 MUSIC 2,O,NO(A),0
1020 MUSIC 3,O,NO(A),0
1030 PLAY C,0,E,D
1038 CLS
1039 PLOT 16,13,10:PLOT 17,13,NO$(A)
1040 PLOT 16,14,10:PLOT 17,14,NO$(A)
1050 PLOT 37,26,STR$(O)
1100 GOTO 200
```

Structure du programme synthétiseur :

1re partie (2Ø à 9Ø) : remplissage du tableau de correspondance code ASCII-touche, numéro note, nom de la note.

- 2Ø ôte le beep du clavier
- NO$: tableau des notes
  NO$(1) = do ; NO$(2) = do #
- 45 l'instruction NO(N) = – 1 est indispensable : en appuyant sur une touche du clavier qui ne donne lieu ni à un son, ni à une modification de son, la ligne 1ØØ sortirait en erreur car MUSIC n'accepte pas Ø en troisième paramètre. On teste donc ces touches en leur donnant une valeur négative qui, en ligne 1ØØØ, donnera le son PING et fera revenir en ligne 2ØØ choisir une autre touche.
- 1ØØ initialisation des variables.

O = octave, C = canal, E = enveloppe, D = durée.

2ᵉ partie (2ØØ à 98Ø) scrutation du clavier.

• 2ØØ saisie du clavier au vol et donc retour en 2ØØ tant qu'on n'appuie sur aucune touche.

• 24Ø test d'arrêt par touche [ESC]

• 4ØØ à 51Ø test de modification d'octave par touches [←] et [→]

• 6ØØ à 71Ø test de modification de l'enveloppe par touche [RETURN], le choix étant F = 1 ou 5 et E valant 1 en début de programme.

• 8ØØ à 91Ø test de modification du canal par touche 1 à 7, le programme n'utilisant que les canaux 1,2 et 3, mixage de 1 et 2.

3ᵉ partie (1ØØØ à fin) : musique.

• 1ØØ5 à 1Ø5Ø : affichent la note jouée au milieu de l'écran en double hauteur et l'octave choisie en bas à droite.

• 1Ø39 et 1Ø4Ø : le 3ᵉ paramètre de PLOT permet d'afficher la note en double hauteur et il est indispensable d'écrire la ligne 1Ø4Ø, répétition de la ligne 1Ø3Ø. (Ce point est traité en détail au chapitre couleur et caractères double hauteurs).

Testez votre ouïe

Vous avez certainement subi un tel test lors du contrôle médical. Vous savez, on vous met des écouteurs et on vous demande si vous entendez. L'opérateur fait varier la fréquence et le niveau. Et de temps en temps (surtout pour les enfants) il supprime le son pour voir si on ne triche pas.

Ce programme fera exactement la même chose. Mais ce sera plus complet car ATMOS ne se fatigue pas. Et à la fin, il vous dira si vous entendez bien ou non.

Nous vous conseillons de brancher ATMOS sur votre chaîne HIFI et d'utiliser un casque.

Lancez le programme. Soyez patient, vous avez 31 tests à effectuer. Et ne trichez pas, ATMOS veille, vous risquez d'être contraint de recommencer.

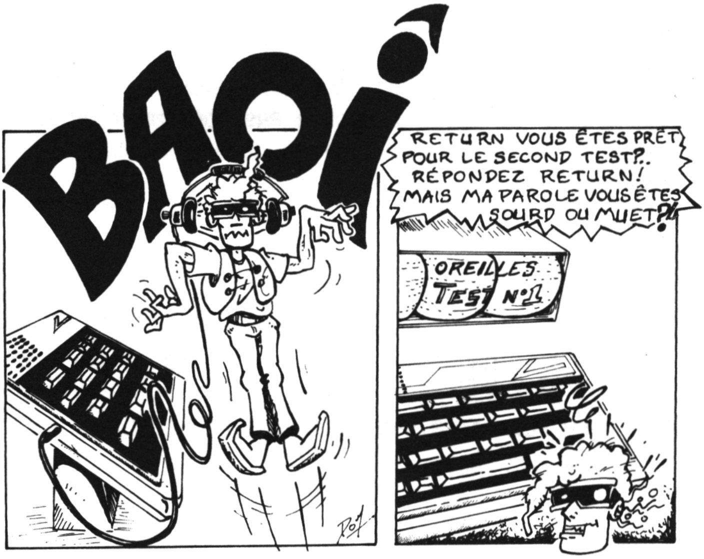
```
5 REM+++OREILLE+++
10 CLS
15 PLOT 4,12,"SINON APPUIE                    SUR UNE AUTRE TOUCHE"
20 LORES0 :DIMT(35):DIMV(35):DIMR(35)
30 DATA 5000,8,2000,8,100,5,100,1,50,5
35 DATA 20,5,10,5,8,8,50,1,20,1
40 DATA 5000,4,2000,3,2000,1,8,1,7,5,7
45 DATA 2,6,9,6,5,6,3,5,7,4,9
46 DATA 14,15,10,25,11,8,27,31,13,24,21
47 DATA 20,19,23,5
48 DATA 22,30,1,16,7,6,26,3,17,9,18,12
49 DATA 28,2,4,29
50 DATA 22,30,1,16,7,6,26,3,17,9,18,12
51 DATA 28,2,4,29
56 DATA 14,15,10,25,11,8,27,31,13,24,21
57 DATA   20,19,23,5
60 FOR I=1 TO 10
70 T(I)=0:V(I)=0:NEXTI
```

```
80 FOR I=11 TO 31
90 READ T(I),V(I):NEXT I
101 FOR N=1 TO 31
110 READ I
120 LORES0:PLAY 0,0,0,0:WAIT 100:PING
130 PLOT 1,10,19:PLOT 2,10,4:PLOT 3,10,12
140 PLOT 4,10,"SI TU ENTENDS APPUIE SUR LA BARRE"
143 PLOT 1,12,20:PLOT 2,12,3:PLOT 3,12,12
145 PLOT 4,12,"SINON APPUIE SUR UNE AUTRE TOUCHE"
150 SOUND 2,T(I),V(I)
160 PLAY 2,0,0,0
170 GOSUB 1000
180 R(I)=R
200 NEXT N
300 REM TRAITEMENT DES RESULTATS
305 S1=0:S2=0:S3=0
310 FORI=1 TO 10:S1=S1+R(I):NEXTI
320 FORI=11TO 20:S2=S2+R(I):NEXTI
330 FORI=21TO 31:S3=S3+R(I):NEXTI
350 PAPER 4:INK 3:TEXT:CLS
360 IF S1>3 THEN PLOT 1,10,"IL N'Y A PAS FORCEMENT UN SON:RECOMMENC
E"
364 WAIT 200
365 IF S1>3 THEN 101
370 IF S1<=3 AND S1>1 THEN PLOT1,12,"TU ENTENDS TROP BIEN,COMME JEA
NNE D'ARC"
375 WAIT 100
380 IF S1>1 AND S3>=10 THEN PLOT 1,12 "MAIS CE N'EST PAS MAL QUAND
MEME"
390 IF S1<2 AND  S3>10 THEN PLOT 1,10,"C'EST PARFAIT:TU ENTENDS TRE
S BIEN!"
400 IF S1<2 AND S3>8 AND S3<=10 THEN PLOT 1,10,"C'EST BIEN"
410 IF S1<2 AND S3<9 AND S3>6 THENPLOT 1,10,"ATTENTION!TU ENTENDS M
AL"
420 IF S3<6 THEN PLOT 1,10,"VA VITE VOIR LE MEDECIN"
500 END
1000 REM  SAISIE DU CLAVIER
1005 R$=""
1010 REPEAT
1020 R$=KEY$
1025 UNTIL (R$<>"")
1030 IF R$=CHR$(32) THEN R=1:RETURN
1040 R=0    :RETURN
```

Commentaires du programme

Le cœur du programme est aux lignes 150 et 160. En 150, SOUND crée un son qui a une période T(I) et un volume V(I) T(I) et V(I) sont deux tableaux qui ont été remplis au préalable par les lignes 60 et 90 à partir des DATA des lignes 30 et 40.
Remarquons que les 10 premières valeurs sont nulles. Cela signifie que 10 des sons n'existent pas.
Les sons sont classés par difficultés croissante. (Voir les DATA).
Pour ensuite donner les sons dans le désordre, on puise les indices I dans les DATA des lignes 50 et 51. Grâce à la ligne 110 et la boucle 100 à 200.
Remarquez la ligne 130 qui donne un affichage coloré et clignotant : bel exemple d'application du chapitre 2.
Le sous-programme 1000 lit le clavier. Il renvoie dans la variable R la valeur 1 si on appuie sur la barre d'espacement (ligne 1030) et 0 si on appuie sur une autre touche.

Le tableau R(I) enregistre le résultat. Les lignes 300 à 500 permettent d'analyser les résultats. Le tableau R(I) est divisé en trois :
Pas de sons
Sans fautes
Sans difficultés
En analysant les résultats des trois catégories, on pourra savoir si le sujet a triché ou dans le cas contraire s'il entend bien ou mal.
Si le test vous semble trop facile, vous enlèverez un niveau à tous les sons en ajoutant en 95 V(I) = V(I) – 1.



CHECK-LIST

SOUND 209
MUSIC 210
PLAY 211
EXPLODE 209
ZAP 208
SHOOT 208
PING 209

TRAINING

• En l'honneur d'ATMOS, faites-lui jouer le "GOD SAVE THE QUEEN". Utilisez un tableau pour représenter les notes.
• Si le résultat vous déçoit introduisez des pauses.

Premiers pas en langage machine

Qu'appelle-t-on langage machine ?

Ce chapitre s'adresse à ceux qui veulent aller plus loin dans la compréhension du fonctionnement du micro-ordinateur. Pour cela, il faudra pénétrer au cœur même de l'ordinateur, dans ce que l'on appelle le microprocesseur.
Ne vous affolez pas, cela n'a rien de compliqué : vous retrouverez les principes que nous avons essayés de vous inculquer lors des précédents chapitres. Mais la programmation en langage machine est plus contraignante car les instructions sont moins parlantes et surtout vous ne disposerez d'aucune protection. La moindre erreur est généralement fatale : entendons-nous bien, fatale pour le programme, pas pour l'ordinateur et encore moins pour le programmeur.
Nous ne prétendons pas vous faire un exposé complet sur la programmation du microprocesseur 65Ø2 ; nous souhaitons seulement vous donner les notions de base qui vous permettront d'écrire de petits programmes pour ensuite, si cela vous tente aborder du travail plus sérieux par exemple en langage assembleur.



Langage évolué - langage machine

Nous vous avons déjà dit que le cerveau d'ATMOS est un microprocesseur. C'est un circuit électronique très complexe : Il est capable d'effectuer un certain nombre d'opérations arithmétiques et logiques sur les nombres contenus dans les cases de la mémoires d'ATMOS ou dans ses mémoires internes que l'on appelle les registres.

Lorsque ATMOS travaille, c'est le microprocesseur qui effectue toutes les opérations. Mais le microprocesseur est incapable de comprendre les programmes que vous écrivez en BASIC. Il ne comprend que son langage à lui : c'est ce que l'on appelle le langage machine. On a d'ailleurs tort de dire le langage machine car il y en a autant que de machine c'est-à-dire que de microprocesseurs : un 65Ø2 ne se programme pas comme un 68Ø9 ou un Z8Ø.
C'est une des raisons pour lesquelles on a créé des langages évolués (BASIC, FORTRAN, PASCAL, LSE...) qui assurent en principe la portabilité du langage d'une machine à l'autre.
Il faut donc traduire le langage évolué en langage machine. C'est le rôle de l'interpréteur ou du compilateur.
Dans le cas de l'ATMOS il s'agit d'un interpréteur. C'est un programme qui traduit votre programme BASIC ligne par ligne en langage machine au fur et à mesure de l'exécution du programme. Ce programme est en permanence dans la mémoire d'ATMOS. C'est lui qui occupe une grande partie de la ROM. Dès que vous mettez la machine en route, l'interpréteur est prêt à traduire les programmes que vous écrivez en BASIC.

Intérêt du langage machine

Alors pourquoi se fatiguer à apprendre ce langage plus compliqué et moins universel ?
— D'une part, travailler en langage machine impose de savoir exactement comment fonctionne l'ordinateur. Vous verrez comment sont traités les nombres, comment s'effectuent les échanges entre mémoire et microprocesseur, etc.
— D'autre part, cela vous permet de réaliser des choses qui ne sont pas possibles en BASIC.
— Enfin les programmes écrits en langage machine sont plus concis (économisent de la mémoire) et surtout beaucoup plus rapides à l'exécution. C'est assez normal, l'interpréteur BASIC traduit au fur et à mesure vos instructions et cela prend beaucoup de temps.
Et bien courage et vous verrez que cela peut aussi être très amusant.

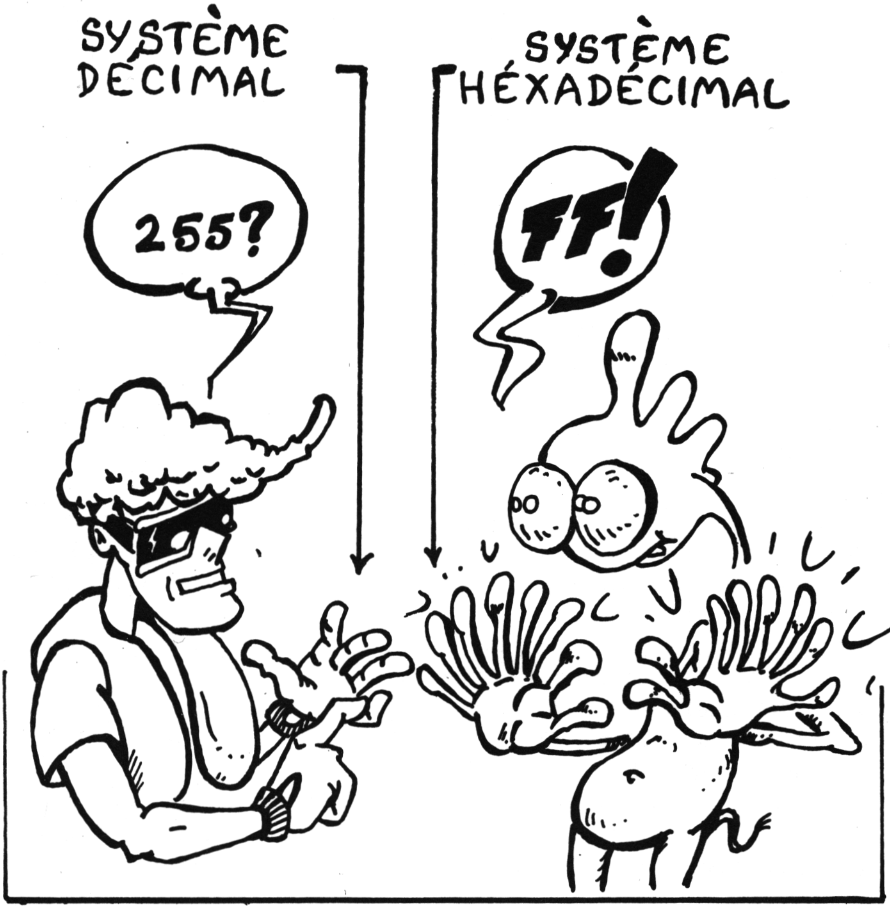
Un peu d'arithmétique

Retour au binaire

Nous avons vu au chapitre "Dans la mémoire" que la mémoire était formée de cases qui contenaient des nombres toujours inférieurs à 256. C'est parce que ces nombres sont mémorisés par des circuits électroniques qui ne peuvent travailler qu'avec deux positions. Les nombres sont par conséquent écrits en base 2 à l'aide de 8 chiffres binaires. On les appelle des octets.

Base 2 ou base 16 ?

Les nombres en base 2 (on dit binaires) sont directement utilisables par le microprocesseur. Par contre nous les trouvons très lourds à manipuler en raison du grand nombre de chiffre qu'ils comportent.
La conversion en décimal d'un nombre écrit en binaire n'est pas très simple et c'est une des raisons pour lesquelles on préfère utiliser la représentation en hexadécimal.
Rappelons les règles de cette représentation.

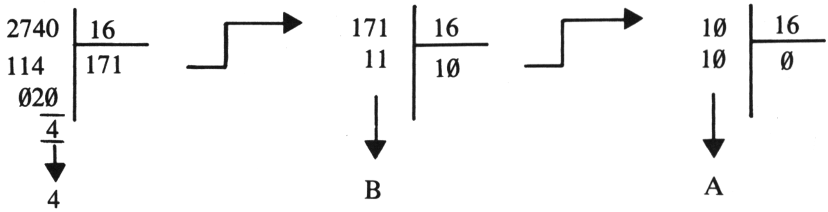


Conversation décimal hexadécimal et inverse.

En hexadécimal, on a besoin de 16 chiffres. On utilise les chiffres décimaux (Ø à 9) avec en plus les six premières lettres de l'alphabet (A à F). D'où la correspondance pour les seize premiers nombres :

Décimal	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15
Hexa	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Pour convertir un nombre décimal en hexadécimal, on utilise des divisions successives. Nous ne vous justifierons pas la règle ; nous vous donnons simplement un exemple.
Soit convertir 274Ø en hexadécimal :
On divise par 16 le nombre puis les différents quotients successifs tant qu'ils ne sont pas nuls.



Hexa :
Le nombre est constitué de la suite des restes convertis en hexa, en commençant par la fin.
Soit : A B 4
Mais ATMOS peut bien sûr faire le travail à votre place ; il suffit d'écrire :
? HEX$(274Ø)
#AB4
Je vous renvoie aux fiches de référence pour plus de précision. Notez la présence du # qui signifie que le nombre représenté est écrit en hexadécimal. Sur d'autres machines on utilise un autre symbole : le H avant ou après ou bien encore le $ devant le nombre.
L'opération inverse (Hexa Déci) est plus simple, il suffit d'appliquer la règle déjà vue à propos du binaire. Par exemple :
4A en hexa signifie $4*16+1Ø$ c'est-à-dire 74
245 en hexa signifie $2*16^2+4*16+5$ soit 581
FA en hexa signifie $15*16+10$ soit 250
C'est simple mais aboutit rapidement à des calculs ennuyeux. Là aussi l'ordinateur vous aidera : il représente les nombres en base 16 avec un # devant mais à l'affichage il les convertit directement en décimal. Exemple :
? #FA
250

Conversion binaire hexadécimal et inverse

Elle est beaucoup plus simple que les précédentes, cela justifie en partie l'utilisation de l'hexadécimal qui n'est en fait qu'un binaire "déguisé"
Pour les seize premiers chiffres :
Il veut mieux les connaître. Sinon vous consulterez le tableau qui suit.

DECI	HEXA	BINAIRE
Ø	Ø	ØØØØ
1	1	ØØØ1
2	2	ØØ1Ø
3	3	ØØ11
4	4	Ø1ØØ
5	5	Ø1Ø1
6	6	Ø11Ø
7	7	Ø111
8	8	1ØØØ
9	9	1ØØ1
1Ø	A	1Ø1Ø
11	B	1Ø11
12	C	11ØØ
13	D	11Ø1
14	E	111Ø
15	F	1111

Si vous n'avez pas le tableau sous les yeux, vous l'écrirez rapidement colonne par colonne.

Pour les nombres quelconques

Je vous donne simplement la règle à utiliser. Prenons un nombre binaire usuel quelconque : 11Ø1Ø11Ø1ØØØ1

On isole des groupes de quatre BITS en partant de la droite.

ØØØ1 1Ø1Ø 11Ø1 ØØØ1 On remplace chaque groupe par le nombre hexadécimal correspondant.

1 A D 1

Le nombre exprimé en base 16 est donc 1AD1

Regardons par exemple quelques valeurs possibles pour un octet :

Binaire	Hexa	Déci
ØØØØØØØØ	Ø	Ø
1111ØØØØ	FØ	24Ø
11111111	FF	255

Vous constatez qu'un octet peut être représenté par deux chiffres hexadécimaux.

La conversion hexa binaire se fera d'une façon similaire :

DE6C s'écrira : 11Ø1111ØØ11Ø11ØØ en binaire.

Nombres négatifs

Le microprocesseur aura parfois besoin de travailler avec des nombres négatifs.

Revenons à la représentation décimale qui nous est le plus familière. Le nom-

bre négatif – 31 est le nombre qui, ajouté au nombre 31 donne zéro, élément neutre de l'addition. La notation utilisée en décimal (mettre un signe devant) est complètement arbitraire. Elle ne permet pas de calculer avec ces nombres. En effet quand on ajoute le nombre – 31, on sait qu'il faut en réalité retranché 31. Cette représentation ne pourrait pas convenir au microprocesseur qui ne pourrait pas calculer avec.

Essayons de trouver autre chose. Supposons un ordinateur qui travaille avec des nombres décimaux de deux chiffres. Quel est le nombre qui ajouté à 31 donnerait zéro ? La réponse est simple, c'est 69. Essayez : 31 + 69 = 1ØØ et si l'ordinateur ne dispose que de deux chiffres, le 1 du 1ØØ est perdu et le résultat sera ØØ vous pouvez en essayer d'autres : – 1 s'écrira 99, – 49 s'écrira 51. C'est ce que l'on pourrait appeler la représentation en complément à 1Ø : Chaque chiffre du nombre négatif, ajoute au chiffre correspondant du nombre donne 1Ø si l'on tient compte de la retenue :
Voici le résultat obtenu :

Décimal	- 50	- 49	- 48	...	- 1	Ø	1	2	3	...	48	49	50	51	...99
Complément à dix	5Ø	51	52	...	99	Ø	1	2	3	...	48	49	X	X	... X

Vous constaterez qu'avec deux chiffres décimaux on représente les nombres de Ø à 99 en représentation normale alors qu'en représentation en complément à 1Ø on pourra écrire les nombres de – 50 à 49. Les nombres négatifs sont ceux dont le premier chiffre est supérieur à 4.
Le microprocesseur utilise pour les nombres négatifs le principe de cette représentation mais avec des nombres bianires de 8 bits c'est-à-dire des octets. Voyons un peu ce que cela donne.
Le premier bit donnera le signe comme précédemment. Mais il n'y a que deux possibilités : pour ne pas contredire la notation habituelle, on conviendra de le mettre à Ø pour les nombres positifs et à 1 pour les nombres négatifs. On l'appelle le BIT de signe.
Le plus grand nombre sera donc Ø1111111 soit 127.
Pour les nombres négatifs, on parlera maintenant de représentation en complément à deux. Comment l'obtient on ?
Reprenez l'exemple du décimal : on remplaçait chaque chiffre par son complément à 9 et on ajoutait 1 au nombre. En binaire on inverse la valeur de chaque BIT et on ajoute un au nombre ainsi trouvé.
Exemple :
86 s'écrit Ø1Ø1Ø11Ø
on
inverse 1Ø1Ø1ØØ1
+ 1
= 1Ø1Ø1Ø1Ø QUI EST la représentation en complément à deux de – 86

Pour vérifier ajoutons les deux nombres :

```
      Ø1Ø1Ø11Ø  86
+     1Ø1Ø1Ø1Ø  -86
=    1ØØØØØØØØ  1ØØ
```

Mais le 1 est perdu car le nombre ne comporte que 8 bits.
Ces nombres (en complément à deux) seront en réalité représentés en hexadécimal en suivant la règle habituelle.
Ainsi 86 s'écrira 56 en hexa et −86 donnera AA
On peut représenter tous les nombres représentables sur 8 BITS en complément à deux.

Déci	−128	−127	...	−2	−1	Ø	1	2	...	126	127
Hexa	8Ø	B1	...	FE	FF	Ø	1	2	...	7E	7F

Comme vous pouvez le constater un octet peut représenter les nombres de −128 à 127 alors que avec des nombres positifs on allait de Ø à 255. Dans la suite, nous ne ferons pas beaucoup de programmes d'arithmétique mais vous utiliserez les nombres négatifs lors des branchements relatifs. Vous aurez alors quelques petits calculs à faire.
Pour conclure, vous savez que ATMOS met un # devant les nombres hexadécimaux. Si cela ne pose aucun problème dans une utilisation courante, c'est très génant en langage machine où le # a en général une autre signification.
Dans toute la suite de ce chapitre, nous utiliserons pour les chiffres hexadécimaux des programmes en langage machine le $.
Par contre, dans les programmes en BASIC, nous continuerons à employer le #.

Programme-instruction-registre

Un premier petit programme

Vous savez qu'un programme en langage machine est une suite de nombres. Ces nombres sont rangés en mémoire dans une suite d'octets successifs. Voici un exemple, les nombres sont écrits en hexadécimal.
A9 41 8D B1 BE
C'est un programme qui affiche un A sur l'écran. C'est le programme tel qu'il se présentera lorsqu'on l'aura chargé en mémoire dans les octets d'adresses $4ØØ à $4Ø4.
C'est évidemment peu parlant aussi préfère-t-on utiliser, pour l'écriture des programmes ce que l'on appelle les mnémoniques. Le programme devient alors :
LDA #$ 41
STA $B6B1

Rappelons que les $ indiquent qu'il s'agit de chiffres hexadécimaux. Ne tenez pas compte du # qui vous sera expliqué plus tard.
Vous constatez que A9 a été remplacé par le mnémonique LDA qui signifie load A c'est-à-dire charger le registre A. BD devient STA qui vient de store A c'est-à-dire ranger le contenu du registre A.
Les autres nombres n'ont pas de mnémonique. Remarquez l'inversion des deux derniers nombres. Toutes ces observations vont trouver une réponse dans ce qui suit.

Qu'est ce qu'un REGISTRE ?

Pour travailler, le microprocesseur a besoin d'un peu de mémoire interne pour stocker des données et faciliter leur traitement. Cette mémoire est en général assez réduite ; elle est formée de cases mémoires de capacité 8 ou 16 bits que l'on appelle les registres. Ces registres n'on pas d'adresse puisqu'ils ne font pas partie de la mémoire (ils sont à l'intérieur du microprocesseur). Par contre ils portent un nom. Pour l'instant, nous ne parlerons que du registre A que l'on appelle aussi Accumulateur. C'est lui qui sera utilisé pour toutes les opérations arithmétiques et logiques. C'est un registre de 8 bits comme les cases de la mémoire.

Contenu d'une instruction.

Reprenons notre petit programme. Nous avons dit que les deux premiers octets (A9 41) sont équivalents à LDA #$41. On distingue dans un programme deux types de nombres. Dans notre cas A9 correspond à un mnémonique. C'est lui qui indique au microprocesseur le type d'opération qu'il doit effectuer. On l'appelle le Code Opération ou code objet. Dans notre exemple, le nombre A9 signifie LDA c'est-à-dire mettre dans le registre A un nombre. Ce nombre est ici précisé dans l'octet qui suit le code opération. On l'appelle l'opérande. Plus généralement, l'opérande fournit les données quantitatives nécessaires à la réalisation de l'instruction. Pour le microprocesseur 65Ø2, le code opération est toujours formé d'un seul octet. Il est suivi de 2 ou 1 octet ou dans certains cas il se suffit à lui-même. Lorsque le code objet est suivi de 2 octets (dans l'exemple précédent, 8D est suivi de B1 et BE), les 2 octets représentent une adresse et l'opérande dépend de cette adresse. Ici l'adresse est $BEB1 et l'opérande est le contenu de cette adresse.
L'adresse est écrite sur deux octets BE et B1 :
B1 est appelé octet de poids faible et BE octet de poids fort (en anglais Most signifiant byte et least signifiant byte). Cette appellation vient de la représentation des nombres : si on l'exprime à l'aide des puissances de 16, l'octet de droite contient les puissances les plus faibles. Voici ce que cela donne :

$$\$BEB1 = \underbrace{11*16+14*16}+\underbrace{11*16+1}$$

soit 48640 + 177 = 48817

On met ainsi en évidence la prédominance du premier octet. Attention à l'écriture : avec les mnémoniques, on écrit :
STA $BEB1 mais en langage machine, cela devient :
8D B1 BE. L'octet de poids faible suit le code opération.
Ne cherchons pas de justification ; sur d'autres microprocesseurs, c'est le contraire.
Remarquez que dans cette représentation, le nombre de 16 bits est vraiment représenté comme deux octets séparés ce qui n'est pas la même chose. Mais le microprocesseur interprétera correctement cette représentation.

Code opération ou opérande ?

Une question se pose ? Comment la machine distingue-t-elle un code opération du nombre qui le suit ? La réponse est simple : c'est le code opération qui doit indiquer au microprocesseur à combien d'octets il s'adresse. Ce dernier possède un registre spécialisé appelé compteur ordinal : PC. Le registre PC contient l'adresse du code opération de l'instruction qui va être exécutée dès que ce code opération est chargé dans le microprocesseur, il lui permet de savoir combien d'octets suivent le code (1,2 ou Ø) et le compteur ordinal peut alors prendre l'adresse du prochain code opération. Le microprocesseur ne pourra donc jamais confondre code opération et nombre à condition qu'on charge en début de programme le registre PC avec l'adresse et la première instruction du programme.
Le compteur ordinal qui doit contenir une adresse est bien sûr registre à 16 bits. C'est le seul sur le 65Ø2.
Comment alors mettre l'adresse voulue dans le compteur ordinal ?
N'oubliez pas que ORIC fonctionne normalement sous BASIC. Cela se fera par conséquent à l'aide d'une instruction BASIC. Il s'agit de l'instruction CALL suivie de l'adresse du premier code opération du programme. Dans l'exemple, si le programme était situé aux adresses $4ØØ à $4Ø4 vous écrirez CALL #4ØØ (attention au # qui est spécifique au BASIC d'ATMOS) pour lancer le programme.
Le registre PC contient alors #4ØØ. Le microprocesseur charge alors le code A9 qui lui dit que ce code est suivi d'un octet. PC est alors immédiatement chargé avec l'adresse du prochain code objet c'est-à-dire $4Ø2 adresse de BD.
L'instruction A9 41 est alors exécutée et le microprocesseur peut alors aller chercher la suivante puisque PC pointe sur son code objet.

Contenus successifs du compteur ordinal	Adresse	Octet
→	4ØØ	A9
	4Ø1	41
→	4Ø2	BD
	4Ø3	B1
	4Ø4	BE
→	4Ø5	?

Dès qu'il a pris connaissance de ce code, le microprocesseur met l'adresse $4Ø5 dans PC. Et après avoir effectué l'instruction STA $BEB1, il va charger l'octet d'adresse $4Ø5 c'est-à-dire n'importe quoi puisque nous n'y avons rien mis. Et nous aurons de gros ennuis.
Vous concevez la nécessité d'une instruction qui va arrêter le programme au plus exactement le renvoyer d'ou il venait c'est-à-dire du BASIC. Cette instruction correspond au RETURN du BASIC, il s'agit du code $6Ø mnémonique RTS qui veut dire retour de sous-programme.
Le programme s'écrira alors :
A9 41 BD B1 BE 6Ø en hexa

```
ou bien : LDA #$41
          STA $BEB1
          RTS                avec les mnémoniques.
```

Il reste un dernier problème à résoudre : comment mettre ce programme dans la mémoire à l'adresse indiquée ?
Si vous avez bien lu le reste du livre, vous vous rappelez certainement l'instruction POKE (voir chapitre "Dans la mémoire") qui vous permet d'écrire un nombre à une adresse donnée de la mémoire. Pour plus de commodité on met le programme en DATA et on lit on POKE octet par octet à l'aide d'une boucle FOR... NEXT.

```
1Ø  DATA #A9, #41, #BD, #B1, #BE, #6Ø
2Ø  FOR I = #4ØØ to #4ØØ+5
3Ø  READ X : POKE I, X
4Ø  NEXT I
```

avec pour l'exécution du programme binaire :

```
4Ø  CALL #4ØØ
```

Ce programme ne sert à rien mais il nous a permis d'aborder les notions de base et il va nous conduire à des programmes plus intéressants.

Notion d'adressage

Qu'appelle t-on adressage ?

Le programme précédent chargeait le nombre $41 à l'adresse $BEB1 de la mémoire. Essayons maintenant d'écrire un programme qui prenne un nombre à un endroit de la mémoire et l'envoie à un autre. La première instruction du programme doit être modifiée : il ne faut plus charger A avec $41 mais avec le contenu de la case d'adresse $9ØØØ. Pour le microprocesseur c'est une instruction différente et cela se traduit par un code objet différent. C'était A9 et cela devient AD.
On dit que dans les deux cas l'adressage est différent. Que signifie ce mot adressage ? C'est la façon qu'utilise le microprocesseur pour aller chercher la donnée dont il a besoin pour effectuer l'opération.
Dans le premier cas, il prend le nombre $41 : on dit qu'il utilise l'adressage immédiat.

Dans le second, il va chercher le nombre qui se trouve à l'adresse $9ØØØ de la mémoire : il s'agit d'un adressage étendu ce qui signifie qu'il peut le prendre à n'importe quel endroit de la mémoire. Le programme devient alors : AD ØD 9Ø BD B1 BE 6Ø.
Si on veut maintenant écrire le même programme avec les mnémoniques, il faut également préciser le mode d'adressage. On n'utilisera pas un mnémonique différent mais on rajoutera un symbole particulier au nombre qui suit le mnémonique. C'était le rôle du # dans le programme précédent.
Le # indique un adressage immédiat. Pour l'adressage étendu on ne rajoute rien. Le programme s'écrit alors :

LDA $9ØØØ →On charge A avec le contenu de la case $9ØØØ
STA $BEB1 →On charge la case d'adresse $BEB1 avec le contenu de A
RTS →Retour au BASIC

Vous avez certainement remarqué que STA $BEB1 utilise aussi l'adressage étendu puisque le nombre qui suit le code objet est l'adresse où doit être effectué le travail.

Les premiers modes d'adressage

Nous avons déjà décrit l'adressage immédiat et l'adressage étendu. Rappelons brièvement leurs caractéristiques :

Adressage immédiat
Lorsque l'octet qui suit le code objet est l'opérande. Permet de charger un accumulateur avec une valeur numérique. (Toujours la même)

Adressage étendu ou absolu
L'opérande est ici le contenu de la case mémoire dont l'adresse suit le code opération. Cette adresse est codée sur 2 octets ce qui permet d'adresser toute la mémoire.
D'autres modes d'adressage sont possibles : voici les plus simples. Les autres seront évoqués dans la suite du chapitre.

Adressage en page zéro
Comme pour l'adressage étendu, l'opérande se trouve dans la case mémoire dont l'adresse suit le code opération. Mais cette adresse est indiquée par un seul octet. Cela limite l'adressage aux 256 premieres cases mémoire. C'est ce que l'on appelle la page zéro. (On considère que la mémoire est divisée en 256 pages, de 256 octets chacune).
Exemple : LDA $41 s'écrit A9 41 et signifie que l'on charge A avec le contenu de l'adresse $41.

Adressage implicite
Certaines instructions ne comportent qu'un seul octet : le code opération. Ce sont généralement des instructions très particulières qui agissent sur les registres. Vous en connaissez une : RTS que nous avons déjà utilisée. En voici une autre : INX signifie qu'on augmente de 1 le contenu du registre X. Ces instructions n'ont pas d'autres mode d'adressage possible. Elles correspondent à un seul code opération que vous trouverez aisément dans l'annexe 1Ø

Adressage accumulateur
C'est le cas lorsqu'une instruction agit sur l'accumulateur à la place d'une case mémoire. Par exemple l'instruction ROL qui effectue un décalage à l'intérieur d'un octet s'écrit :
ROL $56D4 (2E D4 56 en langage machine) si elle agit sur l'octet situé à l'adresse $56D4
ROL (2A en langage machine) si elle agit sur le contenu de l'accumulateur. Et dans ce cas, elle ne comporte qu'un seul octet.

Branchement, registre d'état, adressage indexé

Retour au BASIC

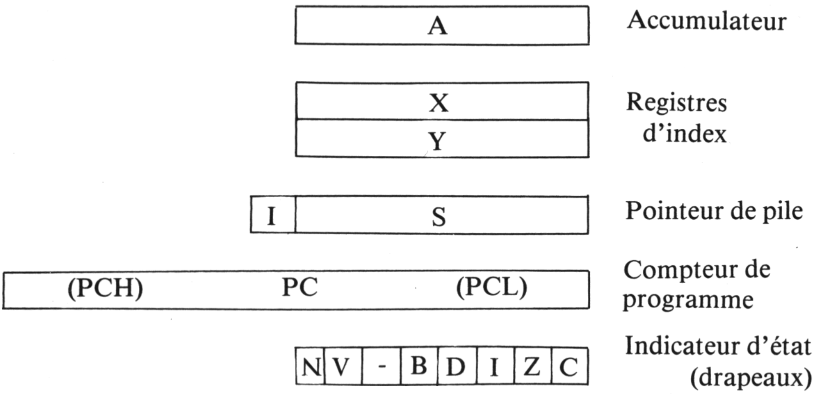
Nous allons démarrer sur un exemple que nous allons d'abord traiter en BASIC. Nous avons vu comment transférer un octet d'une case mémoire à une autre ; nous allons maintenant essayer d'effectuer cette même opération plusieurs fois (par exemple 255) en partant toujours de la même case. Cela se traduira, si on transfère dans la mémoire d'écran par l'affichage de 255 lettres identiques.
Comment traiter le problème ? Il faut :
• Charger le code de la lettre dans une variable à partir de la case mémoire où on doit le prendre (#9ØØØ)
• Transférer le contenu de cette variable à l'aide d'une boucle dans les cases mémoires successives.
Voici une version possible :

```
1Ø  A=PEEK (#9ØØØ)
2Ø  COMPT = 255
3Ø  POKE #BEB1+COMPT,A
4Ø  COMPT = COMPT-1
5Ø  IF COMPT < >Ø THEN 3Ø
```

Nous avons choisi de ne pas utiliser la boucle FOR... NEXT pour nous rapprocher du langage machine. Nous avons réalisé un compteur (COMPT). La ligne 3Ø permet de charger le contenu de A à une adresse variable qui dépend du compteur. La ligne 4Ø décrémente le compteur (diminution de 1), il aurait été possible de faire l'inverse. La ligne 5Ø est un branchement conditionnel : elle teste si COMPT est différent de zéro et dans ce cas elle renvoie au chargement à cette nouvelle adresse. On aura bien de cette façon réalisé 256 fois le chargement à des adresses comprises entre #BEB1 et #BEB1+255.
Sommes-nous capables d'écrire ce programme en langage machine ?
Nous savons effectuer les transferts de mémoire.
Nous ne savons pas le faire avec une adresse variable. Nous allons voir que c'est le rôle de l'adressage indexé.
Nous ne savons pas faire de branchement ni de test. Nous verrons que c'est possible avec les instructions de branchement conditionnelles qui testent les BITS du registre d'état.

Adressage indexé

Registre du microprocesseur 6502



N Signe moins (nombre négatif)
V Débordement
B Break (interruption)
D Décimal
I Inhibition d'interruption
Z Indicateur de zéro
C Indicateur de retenue

Le microprocesseur 65Ø2 possède deux registres supplémentaires que l'on appelle registres d'indexé. Ce sont des registres à 8 bits, on les désignent par les lettres X et Y. Un certain nombre d'instructions permettent de les charger ou de modifier leur contenu.
Ces registres peuvent être utilisés pour mémoriser des données mais leur utilisation est plutôt réservée à l'adressage indexé. Que signifie ce terme ?
Prenons l'exemple du chargement de l'accumulateur. Si vous regardez le tableau de l'annexe, vous constatez que le code opération pour LDA avec l'adressage indexé est BD. Si dans un programme, vous rencontrez BD ØØ 8Ø cela signifie que l'on charge l'accumulateur A avec le contenu de la case mémoire d'adresse $8ØØD + (X). (X) est une notation couramment utilisée pour désigner le contenu de X (ou le contenu d'un autre accumulateur si on change la lettre).
Si le registre X contient la valeur $6A, l'accumulateur A recevra le nombre contenu dans la case d'adresse $8Ø6A. Vous concevez maintenant que l'on puisse, en faisant varier le contenu de X et si l'on sait répéter l'opération, charger un nombre plusieurs fois à des adresses successives.

Il existe plusieurs sortes d'adressage indexé suivant que l'on utilise le registre X ou Y et que l'adresse qui suit est sur un ou deux octets. On parle d'adressage absolu indexé (sur X ou sur Y) et de l'adressage page zéro indexé (sur X ou sur Y). Dans chaque cas, le code opération est différent. Sachez aussi que toutes les instructions ne possèdent pas tous ces modes d'adressage et il vous faudra systématiquement consulter le tableau de l'annexe 1Ø.
Lorsqu'on écrit les programmes avec les mnémoniques, l'adressage indexé sera indiqué par l'adresse suivie de ,X ou ,Y suivant le registre utilisé. On écrira par exemple :
LDA $8ØØØ,X pour l'exemple précédent (Adressage absolu indexé sur X)
LDA $77, X pour un adressage page zéro indexé sur X.
En langage machine, cela donnera :
BD ØØ 9Ø pour la première instruction :
B5 77 pour la deuxième.
Si nous reprenons notre programme, nous constatons que la ligne 3Ø du programme basic correspond exactement à ce type d'adressage si A est le contenu de l'accumulateur. On met le contenu de A à l'adresse #BEB1 + COMPT qui est variable. X jouera donc le rôle de COMPT et on écrira en langage machine :
STA $BEB1,X (Attention aux #du BASIC qui deviennent des $)
Soit en langage machine :
9D B1 BE
La décrémentation du compteur se fera très simplement par l'instruction DEX qui s'écrit sur un seul octet par le code DA et signifie : Décrémentation du registre d'indexe X (Adressage implicite). Il nous reste à voir comment effectuer la répétition.

Branchements conditionnels

Deux parties bien distinctes se trouvent dans cette expression : le branchement et la condition. Nous allons les retrouver dans les instructions correspondantes.

Notion de branchements
On pense bien sûr au GOTO du BASIC. Il s'agit de rompre le déroulement normal des instructions. L'instruction équivalente existe : il s'agit de JMP suivie d'une adresse sur deux octets. Exemple :
JMP $3456 (4C 56 34) signifie que la prochaine instruction qui sera exécutée sera celle dont le code opération se trouve à l'adresse $3456.
En réalité, ce n'est pas un branchement et ce n'est pas conditionnel, c'est un saut. Nous ne l'utiliserons pas ici.
Alors qu'appelle-t-on branchement ?
L'effet est un peu le même mais cela se passe différemment. Il en existe un certain nombre : BEQ, BMI, BNE... Elles comportent toutes un test dont nous parlerons dans le paragraphe suivant et qui diffère pour chacune d'entre elles. Par contre la façon dont se fait le branchement est toujours la même.
L'octet qui suit le code opération indique le nombre d'octets qu'il faut sauter.

Plus précisément, cet octet est ajouté au contenu du compteur ordinal, le registre qui indique l'adresse de la prochaine instruction à exécuter.
Ce n'est pas difficile mais il faut faire très attention. Prenons un exemple :
Voici un début de programme :

LDX #ØØ
STA $5678,X
DEX

Et on veut alors revenir à la deuxième ligne avec par exemple l'instruction BNE.
Nous vous conseillons d'opérer de la façon suivante : vous utiliserez 5 colonnes qui contiendront : l'adresse, le contenu de l'octet, une étiquette, le mnémonique et le reste de l'instruction.
Vous commencez à écrire le programme sur les deux dernières colonnes en tenant compte du nombre d'octets après chaque code objet. Voici ce que cela donne dans notre exemple si on décide d'écrire le programme à partir de l'adresse $AØØØ.

ADRESSE	OCTET	ÉTIQUETTE	MNÉMONIQUE	RESTE
AØØØ			LDX	#5Ø
AØØ1			—	
AØØ2		BOUCL	STA	$5678,X
AØØ3			—	
AØØ4			—	
AØØ5			DEX	
AØØ6			BNE	BOUCL
AØØ7			—	
AØØ8			RTS	

Vous constatez que nous avons mis un repère là ou nous voulions revenir par le branchement (BOUCL). Cela s'appelle une étiquette (ou un label). Ensuite, lorsqu'on veut commander le branchement, on remet l'étiquette après le mnémonique.
Il faut ensuite compléter le tableau en particulier la deuxième colonne celle qui contient le programme en langage machine. Pour cela, vous utilisez le tableau de l'annexe 1Ø pour les codes objets et vous respectez les règles déjà vues pour les adresses.

ADRESSE	OCTET	ÉTIQUETTE	MNÉMONIQUE	RESTE
AØØØ	AØ		LDX	#5Ø
AØØ1	ØØ			
AØØ2	9D	BOUCL	STA	$5678,X
AØØ3	78			
AØØ4	56			
AØØ5	CA		DEX	
AØØ6	DØ		BNE	BOUCL
AØØ7	?			
AØØ8	6Ø		RTS	

Il reste à résoudre le problème du branchement : Quel nombre doit-on mettre après le code DØ qui représente BNE ?
Il faut que ce nombre, ajouté au contenu du compteur ordinal donne l'adresse de l'instruction où l'on veut aller c'est-à-dire 4ØØ2. Or que contient le compteur ordinal ? Méfiance, ce n'est pas AØØ6 car il pointe déjà sur l'instruction suivante. C'est donc AØØ8, adresse de l'instruction RTS. Il faut par conséquent dans notre exemple ajouter -6 (décimal) pour avoir AØØ2. Il suffit de convertir -6 en hexadécimal (rappelez-vous le paragraphe sur l'arithmétique) cela donne $FA. Le nombre qui sera dans la case AØØ7 sera donc FA et votre programme est terminé.
Nous avons découvert un nouveau mode d'adressage : c'est l'adressage relatif : il est relatif à une adresse donnée (celle du compteur ordinal), on lui ajoute un nombre. C'est comme quand vous dites que monsieur Martin habite trois maisons plus loin que monsieur Durand. Le branchement peut être négatif comme vous venez de le voir : cela entraîne un retour en arrière dans le programme. Il peut être positif et on saute alors un certain nombre d'instructions.

Dans tous les cas, vous déterminerez le nombre qui suit le code du branchement avec la méthode que nous venons d'utiliser.

Le test : le registre d'état

Il nous reste à voir comment la machine peut faire des tests sur les registres. Elle n'a qu'un nombre limité de possibilités et elle utilise pour cela des indicateurs. Ces indicateurs n'ont que deux valeurs possibles (il en est de même pour les tests) et ils peuvent être représentés chacun par un bit. Ces bits sont regroupés dans un registre appelé registre d'état ou registre code condition.
Voici la structure de ce registre qui comporte 8 bits : seuls 7 sont utilisés. Chacun porte un nom et est l'indicateur de quelque chose.

N	V	-	B	D	I	Z	C

Celui qui nous intéresse est le Z qui est l'indicateur de ZÉRO. Il est mis à 1 lors de toutes les opérations de chargement de registre ou de mémoire ou lors des opérations arithmétiques ou logiques à condition que le résultat soit nul. Dans ce cas contraire, il est mis à Ø. Pour les autres indicateurs, c'est un peu la même chose, mais ils indiquent une autre particularité (signe, débordement, retenue...). La description complète du rôle de tous les indicateurs, des instructions qui les affectent et de celles qui les utilisent dépasseraient largement le cadre de ce modeste ouvrage.
Nous ne ferons que décrire le rôle de l'indicateur de zéro Z dans l'instruction BNE. C'est très simple : Si Z vaut 1, c'est-à-dire si l'instruction qui précédait se rapporte à un nombre nul le branchement n'a pas lieu et le programme continue normalement. Par contre, si Z=Ø le branchement se fait et le programme "saute" comme indiqué précédemment. BNE signifie Branch if Not Equal.

Écriture du programme

Nous en savons maintenant suffisamment. Il suffit de recoller les morceaux. Écrivons d'abord avec les mnémoniques :
Nous supposons, comme en Basic que l'octet à transférer se trouve à l'adresse $9ØØØ.

```
LDA $9ØØØ             Chargement de A
LDX #$FF              Chargement du compteur X avec 255
BOUCL STA $BEB1,X     Chargement en mémoire à l'adresse $BEB1+(X)
DEX                   Décrémentation du compteur. Affecte éventuelle-
                      ment l'indicateur Z si X est nul.
BNE BOUCL             On recommence si X n'est pas nul
RTS
```

Comme vous pouvez le constatez cela ressemble beaucoup au programme BASIC. Il ne nous reste plus qu'à traduire en langage machine : opérons comme précédemment.

ADR Hexa	OCTET Hexa	ETIQUETTE	MNEMO	RESTE
9ØØ1	AD		LDA	$9ØØØ
2	ØØ			
3	9Ø			
4	A2		LDX	# $FF
5	FF			
6	9D	BOUCL	STA	$BEB1,X
7	B1			
8	BE			
9	CA		DEX	
A	DØ		BNE	BOUCL
B	FA			
C	6Ø		RTS	

Le branchement est de −6 (retour de 6 en arrière) ce qui donne $FA. Le programme est mis en mémoire à partir de l'adresse $9ØØ1 : Il aurait aussi bien pu être mis ailleurs ; c'est un des avantages des branchements relatifs : ils ne dépendent pas de l'adresse absolue. On dira que ce programme est translatable.

Pour l'essayer, il nous faudra le charger à l'aide d'un programme BASIC :

```
 5 CLS
1Ø HIMEM #9ØØØ
2Ø DATA AD, ØØ, 9Ø, A2, FF, 9D, B1, BE, CA, DO, FA, 6Ø
3Ø FOR I= #9ØØ1 TO #9ØØ1+11
4Ø READ X$
5Ø X=VAL("#"+X$) : POKE I, X
6Ø NEXT I
7Ø GET A$ : POKE #9ØØØ, ASC(A$)
8Ø CALL #9ØØ1
9Ø GOTO 7Ø
```

Quelques explications :
La ligne 5Ø rajoute un # à chaque valeur du DATA ce qui évite d'écrire autant de # que de DATA.
La ligne 7Ø saisit un caractère au clavier et met à l'adresse #9ØØØ son code ASCII.
La ligne 8Ø appelle le programme binaire et affiche 255 caractères.
La ligne 1Ø interdit à l'interpréteur BASIC d'utiliser les adresses supérieures à #9ØØØ. (Ce qui détruirait notre programme)
Lancez le programme et appuyez sur une touche. Et recommencez.

Comparaison langage-machine BASIC

Nous venons d'écrire un programme, nous allons l'améliorer pour le rendre plus intéressant. Ce nouveau programme fait la même chose que le précédent mais il remplit tout l'écran. Faute de place, nous ne détaillerons pas les explications ; sachez seulement qu'on utilise une deuxième boucle (Indexée sur Y) dans laquelle la première est imbriquée.

```
LDA #9ØØØ
LDY #5
BOUC 1 LDX #ØØ
BOUC 2 STA $BBA8, X
DEX
BNE BOUC2
INC #$9ØØA
DEY
BNE BOUC1
LDA #$BB
STA $9ØØA
RTS
```

Nous avons intégré ce programme à la suite d'un programme BASIC qui fait la même chose : Remplir l'écran avec des Ø.

```
1 ' ECRAN
2 '
3 '
5 HIMEM#9000
10 CLS
15 FOR I=#BBA8 TO #BBA8+1000
16 POKE I,79:NEXT
17 '
18 '
20 DATA AD,00,90,A0,05,A2,00,9D
25 DATA A8,BB,CA,D0,FA,EE,0A,90
26 DATA 88,D0,F2,A9,BB
30 DATA 9D,0A,90,60
35 '
36 '
```

```
40 FOR I=#9001TO #9001+24
50 READ X$:X=VAL("#"+X$):POKE I,X
60 NEXT I
70 GET A$:POKE #9000,ASC(A$)
90 CALL #9001
95 '
96 '
100 GOTO 70
```

Faites exécuter : Vous voyez d'abord l'écran se remplir de Ø. C'est la ligne 15 du programme BASIC et c'est assez lent. Lorsque c'est fini, apppuyez sur une touche autre que Ø : vous voyez toutes les lettres se transformer INSTANTANÉMENT. Vous avez là une belle démonstration de la rapidité du langage machine.

Si vous n'êtes pas convaincu, voici un autre exemple : vous souvenez-vous du programme RIORIM EMARGORP qui inversait toutes les lettres d'ATMOS ? C'était amusant mais l'opération était très longue. Voici le même écrit en langage machine : vous verrez la différence !

Nous vous donnons directement le programme BASIC. Le programme en langage machine figure dans les DATA. Ce sera un très bon exercice de le retraduire en mnémoniques. Vous pourrez alors essayer de comprendre comment il fonctionne. Ce ne sera pas difficile si vous avez compris ce qui précède. Vous aurez besoin de savoir ce que font les instructions ROR et ROL. Elles effectuent une rotation à gauche (ROL) ou à droite (ROR) sur un registre ou sur une case mémoire.

```
1 '
2 '
3 'RIORIM 2
4 '
10 HIMEM #9000
15 '
20 DATA A2,00,BD,40,B5,A0,09,2A
22 DATA 7E,40,B5,88,D0,F9,CA,D0,F1
30 DATA A2,00,BD,40,B6,A0,09,2A,7E
32 DATA 40,B6,88,D0,F9,CA,D0,F1,60
35 '
40 FOR I=#9001 TO#9001+34
50 READ X$:X=VAL("#"+X$):POKE I,X
60 NEXT
63 '
65 GET F$
70 CALL #9001
75 '
80 GOTO65
```

Faites tourner le programme : il suffit d'appuyer sur une touche pour voir tous les caractères de l'écran s'inverser. Si vous appuyez à nouveau ils redeviennent normaux.

Vous pourrez par exemple retrouver le programme RIORIM EMARGORP pour voir la différence de vitesse.

Une remarque pour finir : si vous faites une erreur dans un programme en langage machine, la machine se bloque car elle interprète comme un code objet un octet qui n'en est pas un. Cela se traduit par un programme qui boucle et vous perdrez le contrôle. Il ne vous reste plus qu'à débrancher et rebrancher (même le RESET n'agit plus). Alors sauvez toujours vos programmes avant le PREMIER essai.

Et l'assembleur ?

Notre but n'était pas de vous faire un exposé exhaustif sur le langage machine du 65Ø2. Il faudrait un livre entier et il en existe de très bons. Nous voulions seulement vous aider à démarrer.
Nous avons délibérément omis certains points importants comme par exemple les sous-programmes, la pile, les instructions arithmétiques, l'adressage indirect...
Nous voudrions simplement vous parlez très rapidement de l'ASSEMBLEUR. C'est un terme dont beaucoup de personnes parlent sans trop savoir ce que cela signifie exactement.
Qu'est ce que l'assembleur ? En réalité cela désigne deux choses.
Un langage de programmation : on devrait dire langage assembleur.
Le programme qui traduit votre programme écrit en langage assembleur en un programme en langage machine.
Comment se présente le langage assembleur ? Eh bien vous en avez presque fait sans le savoir lorsque vous écriviez votre programme avec les mnémoniques et les étiquettes. L'assembleur a en plus quelques instructions spécifiques que l'on appelle directives.
Vous écrirez donc votre programme de cette façon et c'est l'assembleur qui se chargera de traduire en langage machine. Il calculera aussi les branchements et vous dira si vous avez fait des erreurs. C'est quand même beaucoup plus confortable.
Alors si ce chapitre vous a plus et si vous avez tout compris courez vite vous acheter un assembleur.

CHAMP DE FORCE

Et pour clore cet ouvrage, voici un jeu utilisant le langage machine

Au-dessus de vous défile une armada de martiens belliqueux autant que verts. Votre mission consiste à les désintégrer avec votre canon à protons. Seulement voilà... Une barrière magnétique, ultra résistante, s'interpose entre vous et l'objectif, Il vous faudra donc détruire, morceau par morceau, ce champ de force qui ne cesse de tourner sur lui-même et dont les brèches se déplacent sans arrêt. Coup d'œil et astuce sont les seules partenaires sur lesquels vous devrez compter
Pour vous déplacer, vous disposez de deux touches : < pour aller vers la gauche. > pour aller vers la droite. La touche Z vous permettra de tirer

Le temps vous est mesuré pour mener à bien cette mission. Agissez vite si vous voulez réussir à détruire toute l'armée en temps voulu. Le bonus s'additionnera à votre score. Mais attention, votre prochaine mission n'en sera que plus dure. Cela ne saurait vous impressionner, n'est-ce pas ? Gare aux trainards qui laisseront le bonus arriver à zéro

Structure

10-20 Chargement des codes machine et des caractères.
30-60 Initialisations.
100-170 Boucle centrale.
200-220 Déplacement à droite du vaisseau.
300-320 Déplacement à gauche du vaisseau.
500-550 Martien touché.
600 Perdu.
100-1210 Langage machine.
2000-2070 Caractères.

```
1 '
2 '     * CHAMP DE FORCE *
3 '
4 '
5 TEXT:CLS:POKE#26A,10:PAPER0:INK2
9 'Charge car et code machine
10 FORR=0TO40:READA:POKE#400+R,A:NEXT
20 FORR=0TO63:READA:POKE#B508+R,A:NEXT
29 'Initialisations
30 AB=#BE2A:SC=0:AM=#BC9A:TC=0
35 DOKE4,AM:DOKE6,AM-1
40 AV=#BFCC:T=0
45 POKEAV,33:DOKE0,AB:DOKE2,AB-1
50 FORR=AB-1TOAB+37:POKER,34:NEXT
55 FORR=AMTOAM+35STEP5:POKER,40:NEXT
60 DOKE#276,20000
70 PRINT@0,0;"SCORE:";SC:PRINT@23,0;"BONUS:"
100 BO=DEEK(#276):IFBO>60000THEN600
102 PRINT@30,0;BO
105 IFPEEK(#208)=#94THEN200
110 IFPEEK(#208)=#8CTHEN300
115 IFT=1THEN135
120 IFPEEK(#208)=#AATHENT=1:AT=AV-40:SHOOT
130 IFT=0THENCALL#400:GOTO100
135 IFAT<#BBF8THENPOKEAT,32:T=0:GOTO130
140 POKEAT,32:CALL#400:AT=AT-40
150 TS=PEEK(AT)
160 IFTS>33ANDTS<38THENPOKEAT,TS+1:T=0:EXPLODE:GOTO100
165 IFTS=40THEN500
170 POKEAT,39:GOTO100
200 IFAV=#BFDETHEN130
210 POKEAV,32:AV=AV+1:POKEAV,33
```

```
220 GOTO130
300 IFAV=#BFBATHEN130
310 POKEAV,32:AV=AV-1:POKEAV,33
320 GOTO130
500 SC=SC+100:PRINT@7,0;SC
510 T=0:EXPLODE:POKEAT,32:TC=TC+1
520 IFTC<8THEN100
530 SC=SC+DEEK(#276):PRINT@7,0;S
540 AB=AB-40
550 TC=0:POKEAV,32:GOTO35
600 PRINT@31,0;"0":PLOT18,10,"PERDU":END
1000 DATA#A0,#00        'LDY#$0
1010 DATA#B1,#02        'LDA($02),Y
1020 DATA#8D,#FF,#04    'STA$4FF
1030 DATA#B1,#00        'LDA($00),Y
1040 DATA#91,#02        'STA($02),Y
1050 DATA#C8            'INY
1060 DATA#C0,#26        'CPY#$26
1070 DATA#D0,#F7        'BNE$407
1080 DATA#AD,#FF,#04    'LDA$4FF
1090 DATA#91,#02        'STA$($02),Y
1110 DATA#A0,#25        'LDY#$25
1120 DATA#B1,#04        'LDA($04),Y
1130 DATA#8D,#FF,#04    'STA$4FF
1140 DATA#B1,#06        'LDA($06),Y
1150 DATA#91,#04        'STA($04),Y
1160 DATA#88            'DEY
1170 DATA#D0,#F9        'BNE$41C
1190 DATA#AD,#FF,#04    'LDA$4DF
1200 DATA#91,#04        'STA($04),Y
1210 DATA#60            'RTS
2000 DATA30,12,12,45,33,63,63,30
2010 DATA63,63,63,63,63,63,63,63
2020 DATA63,63,47,43,61,55,43,1
2030 DATA63,47,63,45,7,18,0,0
2040 DATA63,58,14,20,0,8,0,0
2050 DATA0,0,0,0,0,0,0,0
2060 DATA8,4,8,4,24,14,28,0
2070 DATA30,51,63,18,33,33,18,18
```

Commentaires

5 POKE #26A, 10 supression bip et curseur.
10-20 Chargement du langage machine et des caractères.
30-45 Initialisation des variables.
50-55 Affichage du mur et des martiens.
60 Mise à 20 000 du bonus.
70 Affichage du score et bonus.
100 Début de boucle centrale : test si le bonus est égal à 0.
102 Affichage du bonus.
105-120 Test clavier.

130 Si aucun missile n'est envoyé, on déplace mur et martiens et on fait le retour de la boucle.
135 Si le missile est arrivé en haut de l'écran, on remet le témoin de tir à zéro.
140-170 Déplacement du missile, du mur et des martiens : test pour savoir si le missile a atteint son objectif.
200-220 Déplacement du vaisseau et test d'arrivée en bout d'écran.
300-320 Idem.
500-520 Martien touché, on ajoute 100 au score et on efface la victime. S'il reste encore des martiens, on retourne dans la boucle principale.
530-550 Tous les martiens sont éliminés, le bonus est additionné au score : le champ de force monte.
600 Perdu.
1000-2070 DATA du langage machine et des caractères avec le code machine en commentaires.

Remarque

Il n'y a pas de routine de décompte du bonus à proprement parler, il se fait tout seul. Pour cela, on utilise des adresses #276 et #277 qui forment, à eux deux, un compteur interne à la machine. Pour l'utiliser, il suffit de taper DOKE #276, X. A partir de ce moment ce qui se trouve en #276 se décomptera tout seul et peut être testé avec DEEK (#276).

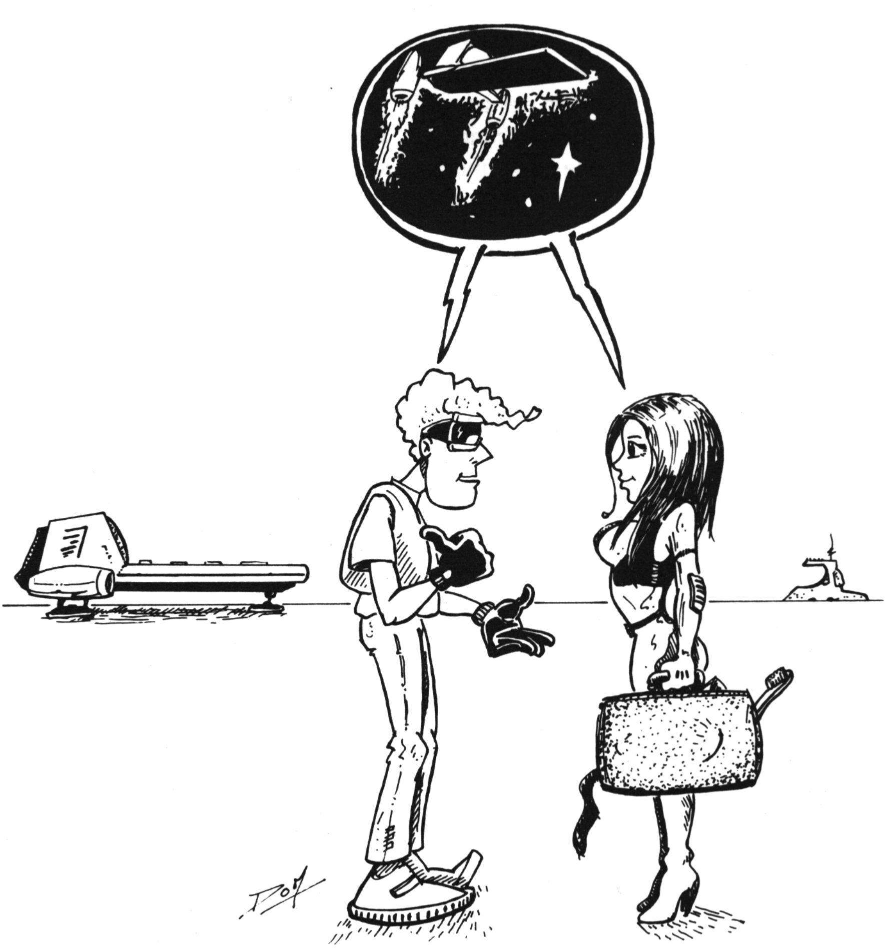
Merci à Gil Espèche, co-auteur de ORIC ATMOS, VOS PROGRAMMES en Basic et langage machine (collection Micromonde, Cedic Nathan) qui a écrit ce programme.

CHECK-LIST

DEEK 217
POKE 214
HIMEM 216
CALL 216
DEFUSR 216
USR 216
DOKE 217
PEEK 216

TRAINING

Il est nécessaire de consulter des ouvrages spécialisés.



Deuxième partie

ATMOS en fiches

Vous voici familiarisé avec le pilotage d'ATMOS. Il vous faut, à présent, acquérir l'expérience qui vous permettra d'effectuer des croisières non-stop aux confins des possibilités de votre machine.
Pour vous y aider voici les fiches de références d'ATMOS. Classées par thèmes (les commandes, les fonctions mathématiques...). Faciles à consulter, elles décrivent l'ensemble des caractéristiques de base d'ATMOS.
Le modèle de présentation est le suivant :

NOM de la donnée, commande, instruction ou fonction.
Syntaxe : donne la ou les façons de l'écrire.
But : indique de façon détaillée le rôle de la donnée, commande, instruction ou fonction.
*Exemple :*court programme ou extrait de programme d'illustration.

Vous trouverez en fin d'ouvrage un lexique Basic ATMOS d'un coup d'œil offrant la liste alphabétique de ces données avec un renvoi à la page de description.

Les commandes

RUN - LIST - NEW - TRON - TROFF - REM - EDIT

RUN

Syntaxe :
RUN

But :
Cette commande lance l'exécution du programme depuis la première insturction.
RUN 5ØØ lance son exécution à partir de la ligne d'instruction 5ØØ.
Cette commande réinitialise les variables numériques à zéro et les variables chaînes à vide.

LIST

Syntaxe :
LIST numéro de ligne

But :
Cette instruction provoque l'affichage en totalité ou en partie du programme présent en mémoire.

Exemple :

LIST 15Ø affiche la ligne 15Ø
LIST 15Ø– affiche toutes les lignes à partir de la ligne 15Ø
LIST –5Ø affiche toutes les lignes jusqu'à 15Ø
LIST 5Ø–15Ø affiche toutes les lignes de 5Ø à 15Ø

NEW

Syntaxe :
NEW

But :
Cette commande efface le programme présent en mémoire vive, met les variables à zéro ou à vide.
On utilise cette commande pour nettoyer la mémoire avant tout nouveau travail afin d'éviter que deux programmes se mélangent.

TRON - TROFF

Syntaxe :
TRON
TROFF

But :
TROFF interrompt l'effet de TRON
TRON
Cette commande a pour effet de visualiser, pendant l'exécution du programme, le numéro des instructions au fur et à mesure de leur exécution. Les numéros apparaissent entre crochet sur l'écran du terminal.
Ce mode permet la détection des anomalies d'un programme en suivant à la trace l'exécution.

Exemple :

TRON : RUN Pour tracer tout un programme

```
100  TRON    Trace le programme entre les lignes 100 et 240
110
120


240  TROFF
```

REM

Syntaxe :
REM<texte>

But :
Cette instruction permet d'insérer des remarques, des commentaires, des titres, etc. afin de le rendre plus "lisible".
Sans effet sur l'exécution du programme, cette instruction permet simplement le maintien de tout ce qui la suit dans le listing.
Il est possible de se brancher par un GOTO ou un GOSUB sur une ligne REM, il y a passage immédiat à la ligne d'instructions suivante.
Il est également possible d'intégrer une ligne de REM dans une ligne d'instructions normale. Dans ce cas, le recours au séparateur ":" est obligatoire.

Exemple :

```
 10  REM+ + +PROGRAMME DE DEMONSTRATION + + + +
100  FOR Y=I TO 1000 : REM Recherche séquentielle
120  T(I)=X$..., etc.
```

EDIT

Syntaxe :
EDIT numéro de ligne

But :
Cette commande de l'éditeur réécrit la ligne 20. Le curseur est positionné en début de ligne. On peut alors effectuer un certain nombres d'opérations :
— recopier la ligne par CTRLA
— modifier la ligne en écrivant
— insérer avec la flèche ←, ↑, ou la flèche ↓
— éviter avec la flèche →
— effacer avec DEL
— valider avec RETURN

Exemple :

```
EDIT 60
60  FOR I=1 TO 400 : NEXT I
```

Entrées et sorties

INPUT - PRINT - TAB - SPC - PLOT - KEY$ - GET - READ - DATA - RESTORE

INPUT

Syntaxe :
INPUT N
INPUT N$
INPUT "chaîne de caractères" ; N,
N$

But :
Cette instruction interrompt le programme en attente de données à introduire. Ces données peuvent être des nombres ou des chaînes.
Fonctionnement :
— le message éventuel "chaîne de caractères" est affiché. Ce message est du genre "votre réponse" ou bien "quantité ?" etc.
— Ce message est suivi d'un point d'interrogation si on a utilisé un point-virgule pour séparer l'instruction (et son message éventuel) de la variable. Si le séparateur est une virgule, le point d'interrogation n'est pas affiché.
— L'utilisateur introduit alors le nombre et le type de données requises par la ou les variable(s). Si ces données ne sont pas du même type, un message REDO demande une nouvelle introduction.
Dans le cas où plusieurs données sont attendues, il faut les introduire en les séparant par une virgule.
— L'utilisateur appuie sur la touche RETURN : les données sont transférées dans les variables et le programme reprend son exécution.

Exemple :
10 INPUT "Nom, Prénom, Age ?", NOM$, PRENOM$, AGE
Après l'affichage du message "Nom, Prénom, Age ?", l'utilisateur entre : DURAND, Paul, 34
Les variables NOM$, PRENOM$, AGE prennent respectivement les valeurs DURAND, Paul, 34.

PRINT

Syntaxe :
PRINT N
PRINT N$
?N
?N$

But :
Cette instruction permet d'afficher, sur l'écran, des caractères alphanumériques.

Exemple :

```
PRINT 5
5
PRINT "BONJOUR"
BONJOUR
10  A$="ADIEU"
20  ?A$
RUN
ADIEU
Ready

10  A$="EST EGAL A"
20  PRINT "TAPEZ UN NOMBRE"
30  INPUT N
40  ?"LE CARRE DE" ; N; A$; N*N
RUN
TAPEZ UN NOMBRE
? 4
LE CARRE DE 4 EST ÉGAL A 16
```

En abrégé, on peut utiliser ? à la place de PRINT, l'ATMOS rétablira PRINT dans les listing.
L'exécution d'une instruction PRINT avec la ligne d'instruction vide provoque le passage à la ligne suivante : très pratique pour aérer votre texte sur l'écran.

Différents modes d'affichage :
Si l'instruction PRINT est suivie d'une liste d'expressions à afficher, celles-ci le sont suivant l'ordre de leur apparition dans la liste avec les variantes ci-dessous :
— un point-virgule après l'expression maintient le curseur sur la même

ligne et provoque l'affichage des valeurs les unes à la suite des autres :

```
10  A=1 : B=7 : C=9
20  PRINT A; B; C
RUN
179
```

— une virgule indique à la tête d'impression qu'elle doit se déplacer jusqu'à la première position de la prochaine zone de tabulation :

```
20  PRINT 1, 7
RUN
17
```

Affichage des valeurs :

Si la ligne de 40 caractères ne suffit pas pour afficher les valeurs d'une liste, l'affichage se fait sur plusieurs lignes consécutives.

Une valeur numérique ne peut être affichée sur plus d'une ligne. En revanche, une chaîne de caractères peut être affichée sur plusieurs lignes consécutives.

Les valeurs positives sont affichées précédées du caractère espace. Les valeurs négatives le sont, précédées du signe moins. Toutes les valeurs numériques sont suivies d'un espace.

Les chiffres réels sont affichés avec au plus six chiffres significatifs.

PRINT (PRINT AT)

Syntaxe :

PRINT X,Y ;

Cette instruction définit les coordonnés d'un caractère dans le mode TEXT et le mode LORES.

X est le nombre des colonnes de 0 à 39 de la gauche vers la droite.

Y est le nombre de lignes de 0 à 27, du haut vers le bas de l'écran.

Ces deux valeurs sont séparées par une virgule et suivies d'un point virgule au-delà duquel commence la liste des éléments à imprimer dans le même format que PRINT.

L'erreur ILLEGAL QUANTITY apparaît si les valeurs fournies dans les coordonnées débordent de l'écran.

L'exemple qui suit trace un cercle qui utilise sinus et cosinus pour calculer l'X et l'Y des positions pour PRINT

```
10  REM PRINT
20  R=13 : XC = 20 : YC=13 : CLS
30  FOR A=0 TO2*PI STEP PI/50
40  X=XC+COS(A)*R
50  Y=YC+SIN(A)*R
60  PRINT  X, Y, "X=X"
70  NEXT
```

TAB

Syntaxe :
PRINT TAB (I)

But :
Cette fonction, complémentaire de l'instruction PRINT, place la tête de lecture sur la colonne dont le numéro est donné par I.

Exemple :
10 PRINT TAB (12), "BONJOUR"
BONJOUR commencera à partir de la douzième colonne en partant de la gauche de l'écran :
...........BONJOUR

Remarques :
TAB ne positionne que les colonnes. Si la position à atteindre se trouve à gauche de la position actuelle, la tête de lecture effectue un saut de ligne, avant d'aller se placer à la colonne désirée.
Le nombre de colonne étant de 40, si I est supérieur à cette valeur, la prochaine position correspond au reste de la division de I par 40.

SPC

Syntaxe :
SPC (X)

But :
Cette fonction crée X blancs lorsqu'elle est utilisée dans une instruction PRINT.
X doit être compris entre 0 et 255

Exemple :

```
10  PRINT "ATMOS" SPC(6) "PREMIER"
20  END
RUN
ATMOS      PREMIER
Ready
```

PLOT

Syntaxe :
PLOT X, Y, "Chaîne de caractères"
PLOT X, Y, N$
PLOT X, Y, N

But :
Cette instruction permet d'afficher en un endroit donné de l'écran un caractère ou une chaîne de caractères.
X définit le numéro de la colonne et Y celui de la ligne. (1 ≤X≤38   Ø≤Y≤26).
Si le caractère est unique, il peut être défini par son code ASCII. Dans le cas où le code ASCII est inférieur à 32, il est considéré comme un ATTRIBUT (Voir annexe) qui modifie l'affichage sur le reste de la ligne.

Exemple :

PLOT 1Ø, 12, "Maison" Affiche Maison sur la 12e ligne à partir de la dixième colonne.
PLOT 1Ø,12,66 Affiche un B (Code ASCII 66) à la case de coordonnées 1Ø et 12.
PLOT 1Ø, 12, 19 Donne un fond jaune sur la ligne 12 à partir de la colonne 1Ø.
1Ø LORES Ø
2Ø PLOT 1Ø,12,19 Affiche un carré jaune de coordonnées 1Ø et 12. (Mode LORES).

KEY$

Syntaxe :
KEY$

But :
Cette fonction sans paramètre "lit" le clavier à la volée.
Si, à cet instant, une touche est enfoncée, elle renvoie la chaîne correspondant au caractère frappé et cela sans avoir à appuyer sur la touche RETURN.
Cette fonction permet d'anticiper l'envoi d'une information au clavier.
Tout caractère de contrôle sauf CNT-C est transmis au programme.

Exemple :
```
1Ø  FOR I = 1 TO 5ØØØ
2Ø  NEXT
3Ø  A$=KEY$
```

40 PRINT A$
RUN
P
Ready
La touche P a été appuyée pendant l'exécution de la boucle FOR.

GET

Syntaxe :
GET X$

But :
Cette instruction arrête l'exécution jusqu'à ce qu'une touche soit actionnée. Elle ne prend qu'un caractère qu'elle affecte à la variable X$.

Exemple :

```
100  GET X$
105  CLS
110  PRINT "Vous avez appuyé sur la touche"; X$
120  GOTO 100
```

A l'exécution, toutes les fois que vous appuyez sur une touche, ATMOS vous le dit.
Le CTRC-C ne fonctionne plus. Prévoir un test de sortie.
Exemple la touche ESC permet de sortir de la boucle.

```
115  IF X$=CHR$(27) THEN GOTO 130
130  END
```

Exemples : La touche ESC permet de sortir de la boucle

READ... DATA

Syntaxe :
READ X
READ X$

But :
Cette instruction lit la donnée indiquée par le pointeur virtuel dans les DATA et l'affecte à la variable (ici X ou X$).

Exemple :

```
10  FOR I=1 TO 6
20  READ N(I)
30  PRINT N(I)
40  NEXT I
50  DATA 110, 78, 6, 45, 34, 89
RUN
110, 78, 6, 45, 34, 89
```

Après exécution :
N(1) vaut 110
N(2) vaut 78
N(3) vaut 6
etc.

Remarques :

L'instruction READ ne peut être utilisée qu'en relation avec un ou plusieurs DATA.
Les types de données employées avec READ doivent correspondre bien entendu aux types de constantes en DATA.

Exemple :

```
10  DATA "Année", 1983
20  READ D,A$
30  PRINT D; A$
40  END
RUN
SN ERROR IN 20
```

Les variables D et A$ doivent être inversées pour correspondre aux constantes 1983 (c. numérique) et "année" (c. chaîne).
Plusieurs instructions READ et DATA peuvent être utilisées dans un programme. Mais l'affectation aux variables commence toujours à la première instruction DATA rencontrée dans le programme.
Un même READ peut servir à "lire" plusieurs DATA. Inversement, plusieurs READ peuvent "lire" une même instruction DATA. Mais lorsque la ligne d'instruction DATA a été lue une fois, la tentative d'exécution d'un nouveau READ provoque la sortie en erreur OD (OUT OF DATA), c'est-à-dire "plus de données".

```
10  DATA 2.5, 1.6, -6.7, 5
20  READ A, B
30  READ C, D, E
40  PRINT A; B; C; D; E;
RUN
OD ERROR IN 30
```

Il y a plus de READ variable que de DATA, le programme s'arrête.
Bien entendu, on peut renouveler les DATA par l'instruction RESTORE.

RESTORE

Syntaxe :
RESTORE

But :
Cette instruction permet de ramener le pointeur de données sur la première donnée en DATA. Après RESTORE le premier READ prend ses valeurs dans le premier DATA du programme.

Exemple :

```
10  READ A, B, C
20  PRINT A, B, C
30  RESTORE
40  READ D, E
50  PRINT D, E
60  DATA 234, 456, 789
RUN
234 456 789
234 456
Ready
```

Après exécution de RESTORE :
— D prend la valeur 234
— E prend la valeur 456

Les fonctions mathématiques

LET - SQR - PI - SIN - COS - TAN - ATN - EXP - LN - LOG - ABS - SGN - RND - INT - DEF FN

LET

Syntaxe :
LET A=5

But :
Cette instruction affecte une valeur à une variable.
LET A=5 : "fais en sorte que A soit égal à 5" On appelle cela une affectation.

Exemple :

```
10  LET E=2.777
20  PRINT E
RUN
2.777
Ready
10  LET B$="SERGE"
20  PRINT B$
RUN
SERGE
Ready
```

Remarques
L'instruction LET est facultative, seul le signe = est impératif. Il va de soi que la variable et l'expression doivent être de même type, sinon sortie en erreur TM.
Si on tape :

```
LET MOIS=12
LET MONT=47
```

et ensuite

```
PRINT MOIS
PRINT MONT
```

on obtiendra chaque fois 47. En effet, seule les 2 premières lettres MO sont lues par votre microodinateur. De sorte qu'en tapant MONT=47, la valeur précédente rangée dans la variable MO(15) a été remplacée.

Autres exemples :

```
100  LET A=12
110  E=12↑2
120  F=12↑4
125  SOMME=12+E+F
130  RUN
20892
Ready

100  A$="GRANDE"
110  B$="ATMOS"
120  C$="MEMOIRE"
130  D$="A UNE"
140  PRINT B$; D$; A$; C$
RUN
ATMOS A UNE GRANDE MEMOIRE
Ready
```

SQR

Syntaxe :
SQR (X)

But :
Cette fonction calcule la racine carrée de X.
X est positif ou nul.
Si X est négatif, il y a sortie en Illegal Quantity Error.

Exemple :

```
10  FOR X=5 TO 35 STEP 5
20  PRINT X, SQR(X)
30  NEXT X
RUN
```

PI

Syntaxe :
PI
Pi est une constante trigonométrique. Elle retourne la valeur 3.145965.
Voici un programme dans lequel PI est utilisé pour calculer la surface d'un cercle :

```
 5  REM PI
10  FOR I=1 TO 5
20  R=RND (1)*30
30  A=PI*(R^2)
40  PRINT "SI LE RAYON DU CERCLE ="R;
50  PRINT "SA SURFACE EST "A
60  PRINT : PRINT
70  NEXT I
```

```
SI LE RAYON DU CERCLE = 21.7104105 SA SURFACE EST
1480.76433
SI LE RAYON DU CERCLE = 9.044449913 SA SURFACE EST
256.991592
SI LE RAYON DU CERCLE = 18.2339642 SA SURFACE EST
1044.50871
SI LE RAYON DU CERCLE = 7.42363369 SA SURFACE EST
173.13423
SI LE RAYON DU CERCLE = 23.486929 SA SURFACE EST
1733.01508
```

SIN

Syntaxe :
SIN(X)

But :
Cette fonction donne le sinus de l'angle X exprimé en radians.

Exemple :

```
10  PRINT SIN (1.5)
20  END
.995495
Ready
```

COS

Syntaxe :
COS(X)

But :
Cette fonction trigonométrique donne le cosinus de l'angle X (l'argument X étant exprimé en radians).

Exemple :

```
10  PRINT COS (– PI/3) ; COS(0) ; COS(PI/2)
→.5 1 0
```

TAN

Syntaxe :
TAN(X)

But :
Cette fonction donne la tangente de X (X étant exprimé en radians).

Exemple :

```
10  PRINT TAN (1.5)
20  END
RUN
14.1014
Ready
```

ATN

Syntaxe :
A = ATN(Y/4)

But :
Donne en radians l'arc tangente.

Exemple :

```
PRINT ATN (1E 38) ; ATN(0) ; ATN(– 1)
1.57080   0   .78540
```

LN2EXP

Syntaxe :
EXP(X)

But :
Cette fonction calcule l'exponentielle e^x avec e = 2.71828.
La valeur de l'argument X doit être inférieure ou égale à 88.Ø 2969 sinon il produit un dépassement de capacité ("overflow") signalé par l'erreur Overflow Error.

Exemple :

```
10  FOR X = .5 TO 2 STEP.5
20  PRINT X, EXP (X)
30  NEXT
RUN
 .5   1.64872
1     2.71828
1.5   4.48169
2     7.38906
Ready
```

LN

Syntaxe :
LN(X)

But :
Fonction logarithme à base e, ou népérien.
Si X < ØIllegal Quantity Error.

Exemple :

```
10  PRINT LN(5)
20  END
RUN
1.60944
Ready
```

LOG

Syntaxe :
LOG (X)

But :
Fournit le logarithme à base dix de X.
Si <Ø Illegal Quantity Error

Exemple :

```
1Ø  PRINT LOG(1) ; LOG (1Ø) ; LOG (2.71828)
Ø   2.8Ø258   1
```

ABS

Syntaxe :
ABS (X)

But :
Cette fonction calcule la valeur absolue de l'argument X. X est une expression numérique de type entier, réel ou double précision.
Le type de résultat est le même que le type d'argument.

Exemple :

```
1Ø  PRINT ABS(7.2) ; ABS(Ø) ; ABS(–.2)
7.2   Ø   Ø.2
```

SGN

Syntaxe :
SGN (argument)

But :
Cette fonction donne –1 si l'argument est négatif ; Ø s'il est nul ; 1 s'il est positif.

Exemple :

```
 5  Y=2 : Z=4
1Ø  X=Y–Z
2Ø  S=SGN(X)
3Ø  PRINT S
RUN
–1
Ready
```

RND

Syntaxe :
RND (X)

But :
Cette fonction RANDOM donne un nombre réel, pseudo-aléatoire (une imitation du hasard) entre Ø et 1.

Exemple :

```
10  FOR I=1 TO 6
20  X=RND (1)
30  PRINT X
40  NEXT
RUN
Ready
```

6 appels successifs de la fonction RND(1) donnent une série de 5 valeurs différentes.
Une nouvelle exécution de ce programme produirait la même séquence.

Remarques :
Avec X = RND(J)
— Si J=Ø le résultat est le même qu'au précédent tirage.
— Si J est inférieur à Ø le résultat sera toujours la même valeur pour un J donné.

Conseil :
Pour obtenir des séquences de nombres aléatoires différentes il faut introduire un nombre entier N et appeler la fonction RND, N fois avant d'entamer l'exécution proprement dite du programme. Exemple :

```
10  INPUT "Donnez un nombre de 1 à 2Ø"; N
20  IF N↘2Ø OR N↘1 GOTO1Ø
30  FOR I=I TO N
40  ALE=RND
50  PRINT ALE (1)
60  NEXT I
RUN
Donnez un nombre de 1 à 20 ? 15
```

INT

Syntaxe :
INT (X)

But :
Cette fonction "partie entière" fournit le plus grand entier inférieur ou égal à la valeur de X.
Le résultat est un nombre réel. L'argument X peut être de type entier ou réel.

Exemple :

```
10  PRINT INT(99.89)
20  END
RUN
99
Ready
10  PRINT INT (−2,5)
20  END
−3
Ready
```

DEFFN FN

Syntaxe :
DEFFN A(X) = Fonction mathématique de X
FNA(X)

But :
DEFFN définit une fonction mathématique qui sera ensuite utilisée par l'expression FN. C'est très utile lorsque l'on utilise une fonction compliquée qui revient plusieurs fois dans un programme.
Le nom de la fonction suit les mêmes règles que celui des variables.

Exemple :

```
10  DEFFN AB(X) = X↑3
20  INPUT X
30  PRINT FN AB(X)
40  GOTO 10
    RUN
    ? 3
    27
    ?2
    8        etc.
```

Fonctions chaînes
LEN - MID$ - RIGHT$ - LEFT$ - ASC - CHR$ - VAL - STR$ - HEX$

LEN

Syntaxe :
LEN (X$)

But :
Cette fonction donne la longuer de la chaîne de caractères X$. Les espaces et les caractères spéciaux sont comptés.

Exemple :

```
10  NOM$ = "DURAND"
20  L = LEN (NOM$)
30  PRINT L
RUN
6
```

La chaîne DURAND contient 6 caractères.
Ready

MID$

Syntaxe :
MID$(X$, N, M)
N et M sont des expressions numériques.

But :
Extrait une sous-chaîne de longueur M à partir du caractère N de la chaîne X$.
N doit être compris entre 1 et 255 et M peut être compris entre 0 et 255. Si M est absent ou s'il est trop grand, la sous-chaîne va jusqu'à la fin de X$; si N>LEN(X$) la fonction donne une chaîne vide. Si N = 0, le message ILLEGAL QUANTITY est détecté, si M = 0 elle donne une chaîne vide.

Exemple :

```
?MID$("TOTO",2)
OTO

?MID$("TOTO",2,2)
OT

?MID$("TOTO",2,34)
OTO

10  A$="ATMOS"
20  B$="EST RAPIDE"
30  PRINT A$; MID$(B$,4,7)
40  END
RUN
ATMOS RAPIDE
Ready
```

RIGHT$

Syntaxe :
RIGHT$(X$,I)

But :
Cette fonction extrait les caractères d'une chaîne X$, en partant de la droite sur une longueur I.

Exemple :

```
10  X$="ATMOS A BEAUCOUP DE MEMOIRE"
20  PRINT RIGHT$ (X$,7)
30  END
RUN
MEMOIRE
Ready
```

Remarques :
I ne doit pas dépasser 255.
Si I est supérieur à la longueur de la chaîne X$ (chaîne origine), la chaîne extraite est la chaîne origine.
Si I=Ø, on obtient une chaîne vide.
Si I est inférieur à Ø, l'erreur FC est signalée.

LEFT$

Syntaxe :
LEFT$(X$,I)

But :
Cette fonction donne une chaîne composée des I premiers caractères de la chaînes X$. I ne doit pas dépasser 255. Si I est plus grand que la longueur de X$, le résultat est X$.

Exemple :

```
10  X$="MICROORDINATEUR ATMOS"
20  Y$=LEFT (X$,15)
30  PRINT Y$
40  END
RUN
MICROORDINATEUR
Ready
```

ASC

Syntaxe :
C=ASC(X$)

But :
Donne sous forme numérique le code ASCII du premier caractère de la chaîne X$
Si la chaîne X$ est vide, il donne un message Illegal Quantity Error.

Exemple :

```
10  X$="BONJOUR"
20  PRINT ASC(X$)
RUN
66 (le numéro de code ASCII de B est 66)
Ready
10  PRINT ASC("2")
50

10  PRINT ASC("?")
63
```

Vous trouverez la liste complète des codes page 225. Mais voici un petit programme qui vous permettra d'obtenir tout de suite cette liste :

```
10  PRINT "TAPEZ LE CARACTERE DE VOTRE CHOIX :"
20  X$=KEY$
30  IF X$ =""THEN 20
40  PRINT ASC(X$)
50  GOTO 10
```

N.B. : La fonction inverse de ASC(X$) est CHR$ (Code ASCII) qui donne le caractère dont on fournit le code.

CHR$

Syntaxe :
CHR$(X)

But :
Fonction inverse de ASC, elle donne un caractère correspondant à la valeur du code ASCII de X (l'argument). Cette valeur doit être comprise entre 0 et 255 sinon il donne un message Illegal Quantity Error.

Exemple :

```
10  PRINT CHR$(66)
RUN
B
Ready
10  FOR I=65 TO 80
20  PRINT CHR$(I)
30  NEXT I
RUN
ABCDEFGHIJKLMOP
```

Elle permet d'effectuer la conversion MAJUSCULES - minuscules :

```
PRINT CHR$ (ASC("A")+32 ....)
a     (se reporter à la table du code ASCII)
```

Exemple :

```
10  INPUT "Ton NOM?";N$
20  FOR I=1 TO LEN(N$)
30  PRINT CHR$ (ASC(MID$(N$,I,1,))+32)
40  NEXT I
RUN
Ton NOM ? ATMOS
atmos
Ready
```

Elle permet, en outre, d'envoyer des caractères spéciaux aux terminaux (écran, imprimante, synthétiseur de son)
?CHR$(7) → active la sonnette
?CHR$(13) → provoque un retour en début de ligne (sans saut de ligne)
?CHR$(1Ø) → provoque un saut de ligne
?CHR$(12) → efface l'écran (équivalent de CLS)
?CHR$(17) → efface le curseur ou le rétablit s'il a été effacé.
Voir Annexe : Caractères de commande.

VAL

Syntaxe :
VAL(X$)

But :
Cette fonction donne la valeur numérique de la chaîne de caractères X$ à condition que celle-ci commence par un nombre.

Exemple :

```
1Ø  X$="1Ø8 Place DE GAULLE"
2Ø  C=VAL (X$)
3Ø  PRINT C
4Ø  END
RUN
1Ø8
Ready
```

Remarque :
Si le premier caractère de la chaîne n'est pas un caractère décimal, un signe + ou −, un espace ou un point, le résultat est égal à zéro.

Exemple :

```
1Ø  PRINT  VAL("67") ;  (VAL("7") ;  VAL  ("GIGI") ;  VAL
("7□□□")
2Ø  END
RUN
67   7   Ø   7
Ready
```

Conseil :
Cette fonction est souvent utile pour effectuer des traitements arithmétiques sur des nombres contenus dans une chaîne de caractères. Pour cela, il faut transformer la suite de caractères correspondante en une valeur numérique.

Exemple :

```
10  INPUT "Donnez votre numéro INSEE" ;A$
20  A=VAL (MID$(A$,2,2))
30  PRINT "Vous avez" ; 83 - A ; "ans"
RUN
Donnez votre numéro INSEE ? 1530782182042
Vous avez 30 ans
Ready
```

STR$

Syntaxe :
STR$(X)

But :
Cette fonction convertit la valeur de X en une chaîne de caractères.

Exemple :

```
10  N$=STR$(78)
20  PRINT N$
30  END
RUN
78
Ready
```

Remarque :
STR$ est la fonction inverse de VAL(X$). Elle permet d'appliquer des fonctions chaînes à des constantes numériques. On l'utilise, entre autres, dans l'analyse des résultats numériques.
Très utile avec PLOT pour afficher la valeur d'une variable numérique qui est inférieur à 32.

Exemple :

PLOT 10, 10, 3 n'affiche rien
PLOT 10, 10, STR$(3) affiche un 3.

HEX$

Syntaxe :
HEX$(X)

But :
Cette fonction convertit la valeur décimale de X en une chaîne de caractères représentant la même valeur en base 16.
Si X n'est pas un entier, la fonction HEX$ prend le plus grand nombre entier strictement inférieur à X puis fait la conversion.

Exemple :

```
PRINT HEX$ (26)
1A
```

Instructions de rupture de séquence

END - STOP - CONT - WAIT - GOTO - IF ... THEN... ELSE - FOR... TO... STEP ... NEXT - REPEAT... UNTIL - PULL - GOSUB ... - RETURN - ON... GOTO - ON ... GOSUB - POP

END

Syntaxe :
END

But :
Instruction arrêtant l'exécution d'un programme.

STOP

Syntaxe :
STOP

But :
Cette commande interrompt l'exécution d'un programme à l'endroit où elle se trouve. Elle peut être utilisée à n'importe quel endroit du programme.
Pour reprendre l'exécution du programme taper CONT.

Exemple :

```
10  INPUT X,Y
20  K=INT(X/Y) : PRINT K
30  STOP
40  L=INT(K/Y)
50  PRINT L
60  END
RUN
?16,2
8
Break in 30
Ready
CONT
4
Ready
```

CONT

Syntaxe :
CONT

But :
Provoque la reprise de l'exécution d'un programme après un arrêt (BREAK) provoqué par l'envoi du caractère CTRL C ou par l'exécution de l'instruction STOP.
L'exécution reprend à l'endroit où elle a été interrompue. Dans le cas où CTRL C aurait été envoyé au moment de la réponse à l'instruction INPUT, lors de la reprise, le texte entre guillemets éventuellement spécifié dans INPUT apparaît à nouveau sur l'écran.

Exemple :

```
100  INPUT "QUEL NOMBRE" ; X
150  STOP
160  PRINT "LA RACINE CARRÉE DE ";X;"EST"; SQR(X)
RUN
QUEL NOMBRE ?
?1234567890

BREAK IN 150
CONT
LA RACINE CARRÉE DE 1.23457E+09 EST 35136.4
```

Remarque :
La commande CONT facilite la mise au point du programme que vous êtes en train d'écrire. Par exemple, si vous avez arrêté l'exécution de votre programme à un endroit déterminé par la commande STOP, vous pouvez tester cette partie du programme ou bien afficher le contenu des variables au moyen de l'instruction PRINT, en mode immédiat. La commande CONT vous permet de reprendre l'exécution à l'endroit où elle avait été interrompue, à condition de ne pas modifier le programme après l'interruption.
En mode bureau (ou mode immédiat), l'instruction GOTO permet de poursuivre l'exécution à un numéro de ligne déterminé.

WAIT

Syntaxe :
WAIT N

But :
Cette commande interrompt l'exécution du programme en cours pendant N fois 10 ms.

Exemple :
200 WAIT 60 arrête l'exécution pendant 600 millisecondes, soit à peine plus d'une demi-seconde.

GOTO

Syntaxe :
GOTO numéro de ligne
GO TO numéro de ligne

But :
Cette instruction entraîne le branchement au début de la ligne indiquée par "numéro de ligne".

Exemple :

```
45  PRINT "JE ME REPETE," ;
50  FOR I=1 TO 1000 : NEXT I
60  PRINT "MAIS JE NE SUIS PAS GATEUX !"
70  GOTO 45
```

(Pour arrêter appuyez sur CNT et simultanément sur C).

Remarque :
Si la ligne désignée ne contient pas d'instruction exécutable (par exemple : REM suivi d'un commentaire), le branchement se fait sur la ligne suivante.
Si le numéro de ligne n'existe pas, l'erreur US (Undefined Statement) est signalée.

Conseil :
GOTO numéro de ligne est l'équivalent de RUN numéro de ligne : ces deux instructions permettent le lancement direct au numéro de ligne indiqué (attention GOTO ne réinitialise pas les variables). On peut également l'utiliser en mode bureau pour poursuivre au numéro de ligne indiqué un programme interrompu.
Ceci est très utile pour la mise au point des programmes.

IF... THEN... ELSE...

Syntaxe :
IF expression
THEN numéro de ligne ou instruction(s)
ELSE numéro de ligne ou instruction(s)

But :
Cette instruction permet des branchements sous certaines conditions. L'exécution se fait ainsi :
— si l'expression suivant IF est vraie ou n'est pas nulle alors les instructions qui suivent THEN sont exécutées ;
— si l'expression est fausse ou si on résultat est nul, le programme exécute la (ou les) instruction(s) après ELSE.
N.B. : ELSE peut être omis ; dans ce cas le programme saute à la ligne suivante si l'expression est fausse.

Exemple :

```
10  INPUT "NOMBRE A, NOMBRE B" ;A,B
20  IF A>B THEN PRINT "A" ELSE PRINT"B"
30  GOTO 10

10  PRINT "QUEL EST VOTRE NOM" : INPUT N$
20  PRINT "ETES VOUS DU SEXE FEMININ";N$;"OUI/NON) ?
30  INPUT R$
40  IF R$="OUI" THEN 60
50  PRINT "BONJOUR MONSIEUR"; N$:END
60  PRINT "BONJOUR MADAME OU MADEMOISELLE";N$
```

La ligne 40 présente un test sur le contenu de la variable R$. Si la réponse est oui, ATMOS exécute immédiatement l'instruction de la ligne 60, sinon il passe à la ligne suivante car R$ = "OUI" est faux et là il s'arrête à l'instruction END.

FOR... TO... STEP... NEXT

Syntaxe :
FOR (variable numérique) = X TO Y [STEP Z]
NEXT (variable numérique)

But :
Cette instruction produit une boucle qui commence à FOR et se termine à NEXT ; elle permet aux instructions situées entre FOR et NEXT de s'exécuter un certain nombre de fois selon le pas (STEP).

— La variable numérique qui suit FOR est appelée variable de contrôle de la boucle. Elle peut être de type entier ou réel mais le type double précision n'est pas accepté (erreur TM).
— L'expression X donne la valeur initiale qui est affectée à la variable de contrôle. Exemple :
FOR I = 1
— L'expression Y donne la valeur finale à ne pas dépasser, au-delà de laquelle on sort de la boucle. Exemple :
FOR I = 1 TO 6
— L'expression Z donne le "pas d'incrémentation" de la variable de contrôle, c'est-à-dire le nombre d'unités dont celle-ci est augmentée, à chaque passage. STEP 1 augmente I de 1 ; STEP 2 l'augmente de 2, etc. Si STEP Z est omise, l'incrément par défaut est de 1. Exemple :
FOR I = 1 TO 6 STEP 2
— NEXT correspond à FOR c'est la limite de la boucle.A toute instruction FOR doit correspondre une instruction NEXT et une seule, mais une instruction NEXT peut correspondre à plusieurs FOR.

Exemple :

```
10  FOR I=1 TO 6 STEP 2
20  PRINT I
30  NEXT I
40  PRINT "C'EST FINI"
RUN
1 3 5
C'EST FINI
Ready
```

Les boucles FOR/NEXT peuvent décompter. Dans ce cas, l'instruction STEP est obligatoire et indique l'incrémentation négative. Exemple :
10 FOR I = 1000 TO 1 STEP – 2

Autre exemple :

```
10  FOR I=10 TO 1 STEP -1
20  PRINT I ;
30  NEXT I
RUN
9 8 7 6 5 4 3 2 1
Ready
```

Boucles imbriquées

Plusieurs boucles FOR...NEXT peuvent être emboîtées les unes dans les autres. Mais il est interdit de les faire se chevaucher. Exemple :

```
10  FOR I=1 TO 5 'Boucle extérieure
20  PRINT "I=" ;I ; "J=" ;
```

```
30  FOR J=1 TO 10 'Boucle interne
40  PRINT J;
50  NEXT J
60  PRINT
70  NEXT I
RUN
I=1 J=1 2 3 4 5 6 7 8 9 10
I=2 J=1 2 3 4 5 6 7 8 9 10
I=5 J=1 2 3 4 5 6 7 8 9 10
Ready
```

Boucle de temporisation

Pour ralentir la vitesse d'affichage d'une liste, il est utile de faire appel à l'instruction PAUSE dans une boucle. Exemple :

```
10  FOR I=1 TO 50
20  PRINT I
30  FOR PAUSE=0 TO 10 : NEXT PAUSE
40  NEXT I
```

Cela revient à faire compter la machine de 0 à 10 à chaque tour de boucle.
N.B. : on peut également utiliser, en guise de "boucle de temporisation", l'expression suivante :

```
30  FOR T=0 TO 10 : NEXT
```

Le T après NEXT n'est pas obligatoire.
Il est encore possible, pour le même résultat, de faire appel à l'instruction WAIT :

```
30  N=100 : WAIT N
```

WAIT arrêtera l'exécution du programme pendant 1 seconde.
WAIT N interrompt l'exécution de N fois 10 millisecondes.

REPEAT UNTIL...

Syntaxe :
REPEAT
UNTIL expression

But :
Les instruction situées entre REPEAT et UNTIL sont répétées TANT QUE le test qui suit UNTIL est faux. A partir du moment où il est vrai, le programme se poursuit à l'instruction qui fait suite à UNTIL.
La boucle s'exécute au minimum une fois.
Une erreur BAD UNTIL sera affichée si aucune instruction REPEAT n'a été rencontrée par l'interpréteur.

Exemple :

```
10  REM SIMULATION D'UN JEU DE DES
20  REPEAT
30  X=X+INT (RND(I)*6)+1
40  PRINT X
50  UNTIL X>100
60  END
```

Remarque :
Les dés sont jetés autant de fois qu'il faut pour que le total des points ne dépasse pas 100. On pourrait obtenir le même type de programme avec IF et GOTO mais avec plus de lourdeur.

PULL

Syntaxe :
PULL

But :
Cette instruction décale d'un cran la pile d'adresse dans les boucles REPEAT imbriquées les unes dans les autres.

GOSUB... RETURN

Syntaxe :
GOSUB (numéro de ligne) (*)
RETURN

But :
Cette instruction permet le branchement d'un programme principal à un programme secondaire (ou sous-programme) situé au numéro de ligne qui suit la première partie de l'instruction. L'instruction RETURN permet le retour au programme principal, qui continue à partir de l'instruction suivant immédiatement l'appel.

Exemple : *sous-programme*

programme principal

```
                                      1000
                                        ↓
100  GOSUB 1000  ──────────────→      1050  RETURN
120  ←─────────────────────────────
300  GOSUB 2000  ──────────────→      2000
310                                     ↓
399  END  ←────────────────────       2100  RETURN
```

Remarques :
Une tentative d'exécution de GOSUB vers un numéro de ligne inexistant est signalé par l'erreur U.S. (UNDEF'D STATEMENT).
Un sous-programme peut appeler un autre sous-programme sans autre limitation que la taille de la mémoire et le respect de la syntaxe d'appel et de retour.
Un sous-programme peut être placé n'importe où dans le corps du programme. Par souci de clarté, il vaut mieux regrouper les sous-programmes en fin de programme principal. De cette façon, ce dernier sera plus lisible.
Si une instruction GOSUB branche le programme principal sur une instruction REM suivi d'un commentaire, l'exécution passe à la ligne suivant la ligne REM.

(*) GO TO SUB ROUTINE : (sub routine : sous-programme en anglais)

ON... GOTO...
ON... GOSUB...

Syntaxe :
ON X GOTO n° de ligne 1, n° de ligne 2, etc.
ON X GOSUB n° de ligne 1, n° de ligne 2 etc.
N.B. : GOSUB ne s'utilise que pour le branchement à un sous-programme.

But :
Ces deux instructions permettent le branchement multidirectionnel :
— selon la valeur de la variable X (ou d'une expression), il y a branchement au numéro de ligne 1, au numéro de ligne 2 etc.

Exemple :

```
10  INPUT "Commande ? 1, 2, 3"; X
20  ON X GOTO 1000, 2000, 2500
30  GOTO 10
1000... : REM Traitement 1
............
2000... : REM Traitement 2
............
2500... : REM Traitement 3
```

Si X vaut 3, il y a branchement en 2500 et début du traitement 3.

Remarque :
L'expression X est évaluée et sa valeur, après conversion à l'entier le plus proche par arrondi.
X<Ø ou X↘255 provoque une erreur (FC)
S'il y a tentative de branchement à une ligne inexistante du programme, l'erreur UL est signalée.

Exemple :

Le programme suivant est un exemple d'aiguillage multiple en fonction d'une réponse.

```
 1Ø  PRINT "Tapez un chiffre entre Ø et 3"
 2Ø  PRINT "APPUYEZ SUR 'BREAK' POUR ARRETER"
 3Ø  INPUT D
 4Ø  IF D-INT(D)<Ø> THEN 14Ø
 5Ø  IF D   Ø THEN 15Ø
 7Ø  REM A PRESENT ON EST SUR QUE D EST Ø,1,2 OU 3
 6Ø  IF D>3 THEN 15Ø
 5Ø  IF D<Ø THEN 15Ø
 8Ø  PRINT "VOUS AVEZ TAPE";D
 9Ø  REM
 95  ON D+1 GOTO 1ØØ, 11Ø, 12Ø, 13Ø
1ØØ  PRINT "ZERO"
1Ø5  GOTO 1Ø
11Ø  PRINT "UN"
115  GOTO 1Ø
12Ø  PRINT "DEUX"
125  GOTO 1Ø
13Ø  PRINT "TROIS"
135  GOTO 1Ø
14Ø  PRINT "NON ENTIER"
145  GOTO 1Ø
15Ø  PRINT "HORS LIMITES"
155  GOTO 1Ø
16Ø  END
```

Lorsqu'un sous-programme suit immédiatement le programme principal il est nécessaire d'éviter l'entrée non voulue dans ce sous-programme par l'instruction END ou STOP. Sinon, l'erreur RETURN WITHOUT GOSUB sera détectée.

Exemple :

```
 10  X=1 : Y=2
 20  GOSUB 100
 30  GOSUB 200
 40  GOSUB 100
 50  END
100  REM Affichage
110  PRINT "X="; X ;" " ; "Y="; Y
120  RETURN
200  REM Permutation des valeurs
210  AUX=X : X=Y : Y=AUX
220  RETURN
RUN :
X=1 Y=2
X=2 Y=1
Ready
```

Le premier sous-programme affiche X et Y. Le deuxième permute X et Y.

POP

Syntaxe :
POP

But :
Cette instruction fait oublier à la machine l'adresse de retour pour l'instruction GOSUB donnée en dernier. Le prochain RETURN rencontré provoquera un branchement juste après l'instruction GOSUB qui précède celle dont l'adresse est oubliée.

Les opérateurs logiques
TRUE - FALSE - NOT - OR - AND

TRUE

Syntaxe :
TRUE

But :
Cet opérateur logique (BOOLEEN) vaut −1 et est l'équivalent de VRAI.

FALSE

Syntaxe :
FALSE

But :
Cet opérateur logique est un booléen. Il a la valeur zéro et correspond à FAUX.

NOT

Syntaxe :
NOT

But :
Cet opérateur logique a la signification de la négation : NOT (A = B) est VRAI lorsque A est différent de B.
Dans l'ordre des priorités des opérateurs logiques, NOT vient en tête.

Exemple :

NOT (4 = 5) vaut −1

OR

Syntaxe :
A OR B

But :
Cet opérateur logique a la signification de la "disjonction non exclusive" : A OU B est VRAI dans les trois cas suivants :
— A VRAI B FAUX
— A FAUX B VRAI
— A VRAI B VRAI
Autrement dit, la condition posée avec OU (OR) est vraie si l'une ou l'autre des propositions écrites de part et d'autre du OU est vraie, ou si les deux sont vraies simultanément.

Exemple :

Dans un calcul de moyennes de notes, on veut vérifier, avant d'entrer dans le calcul, que la note N, tapée par l'utilisateur est bien comprise entre Ø et 2Ø. On écrira :
IF N$Ø OR N£2Ø THEN...

AND

Syntaxe :
A AND B

But :
Cet opérateur logique a la signification de la "conjonction" : A ET B est VRAI dans le cas, et le seul, où les deux conditions A et B le sont simultanément.
Autrement dit chaque proposition de part et d'autre du ET doit être vraie pour que la condition soit vraie.

Exemple numérique :
On veut faire un traitement si le nombre A est compris entre deux bornes, par exemple : 5<A<1Ø. On écrira
IF A <1Ø AND A>5 THEN...
La condition est vraie pour A=6 ou 7 ou 8 ou 9.
La condition est fausse pour toute valeur inférieure ou égale à 5 et pour toute valeur supérieure ou égale à 1Ø.

```
1ØØ  PRINT "1. Consultation/2. Création" ;
     X$=INPUT$(1)
11Ø  IF ASC (X$) < > 49 AND ASC (X$)< >5Ø THEN 1ØØ
12Ø  IF ASC (X$)=49 THEN 1ØØØ ELSE 2ØØØ
.
.
.
1ØØØ 'convertion
2ØØØ 'création
```

Gestion de l'écran
CLS - TEXT - HIRES - LORES - INK - PAPER - POS - SCRN

CLS

Syntaxe :
CLS

But :
Cette instruction efface l'écran et positionne le curseur dans le coin supérieur gauche de la fenêtre.

Exemple :

```
10  REM + + +ESSAI+ + +
20  CLS
30  FOR J=1 TO 20
ETC.
```

CLS s'utilise fréquemment en début ou en cours de programme chaque fois qu'il faut "nettoyer" la fenêtre de travail.

Remarque :
Cette instruction est équivalente à PRINT CHR$(12)

TEXT

Syntaxe :
TEXT

But :
Cette instruction efface l'écran et remet en mode texte. Quand on branche l'ATMOS, il se met automatiquement en mode texte, c'est-à-dire qu'on peut se servir de l'écran pour afficher des lignes de caractères.

Remarque :
Les autres modes de l'ATMOS sont LORES de "LOw RESolution" qui signifie basse définition et HIRES de "HIgh RESolution" qui signifie définition fine. Se reporter à ces rubriques.

HIRES

Syntaxe :
HIRES

But :
Cette commande fait passer en mode haute définition avec fond noir et avant-plan blanc.
Le curseur est placé en Ø,Ø
En bas de l'écran, restent trois lignes de texte qui ne changent pas de couleur.
En mode bureau (ou immédiat), on peut utiliser l'ATMOS comme une table traçante pour tester les instructions graphiques qu'on désire intégrer à un programme.

Exemple :

```
HIRES
CURSET 12Ø, 1ØØ, 1
CIRCLE 5Ø, 1
```

LORES

Syntaxe :
LORES Ø
LORES 1

But :
Cette instruction fait passer en mode basse définition (4Ø×25)
LORES Ø utilise les caractères standard
LORES 1 utilise les caractères semi-graphiques
Le fond devient noir, l'encre blanche.

INK

Syntaxe :
INK N

But :
Modifie la couleur de l'avant-plan pour la totalité de l'écran.
N est un entier de Ø à 7

PAPER

Syntaxe :
PAPER N

But :
Change la couleur du fond sur tout l'écran. N est un paramètre variant de Ø à 7
Rappel du code des couleurs :
Ø Noir
1 Rouge
2 Vert
3 Jaune
4 Bleu
5 Magenta
6 Cyan
7 Blanc

POS

Syntaxe :
POS

But :
Cette instruction retourne la position horizontale du curseur quand elle est utilisée pour détecter une position donnée par PRINT.
Attention, elle ne peut être utilisée pour détecter les positions générées par PLOT ou PRINT AT (PRINT)
POS(Ø) retourne la position du curseur sur l'écran. POS(1) retourne la position sur l'axe horizontal de la tête d'affichage lorsqu'on utilise LPRINT (impression sur imprimante).

```
  5  REM POS
 1Ø  DIM A(4,2)
 15  REPEAT
 2Ø  FOR K=1 TO 4
 3Ø  FOR J=1 TO 2
 4Ø  A(K,J)=K*RND(1)+J
 5Ø  NEXT
 6Ø  NEXT : CLS
 7Ø  FOR K=1 TO 3 : PRINT : NEXT
 8Ø  PRINT A(1,1) ;
 9Ø  REPEAT : PRINT "   " ;:UNTIL POS(Ø)=15
1ØØ  PRINT A(1,2)
```

SCRN

Syntaxe :
SCRN (X,Y)

But :
Permet de connaître le caractère qui occupe le point de coordonnées X,Y sur l'écran. SCRN est l'abréviation de screen (écran).

Exemple :

Si on a effectué la commande PLOT 15,12,"A", la lettre A apparaîtra au point de coordonnées 15, 12. Si l'on tape :
PRINT SCRN (15,12)
Le nombre 65 sera fourni : c'est le code ASCII du caractère A.
Cette commande peut rendre des services dans un jeu pour repérer l'emplacement de caractères définis.

Graphique haute résolution
CURSET - CURMOV - DRAW - CIRCLE - PATTERN - FILL - CHAR - POINT

CURSET

Syntaxe :
CURSET X,Y,FD

But :
Cette instruction place le curseur à la position X, Y, où le point sera marqué.
Ø ≤ X ≤ 239 et Ø ≤ Y ≤ 199
FD est un paramètre dont les codes sont :
Ø Couleur du fond
1 Couleur de l'encre
2 Couleur inversé
3 Pas de couleur

Exemple :

CURSET 12Ø, 1ØØ, 1
place le curseur au centre de la fenêtre de travail et marque un petit point (ou pixel).
Le premier paramètre représente la position horizontale X (ici 12Ø), le second la position verticale Y (ici 1ØØ) FD valant 1 correspond au code "couleur de l'encre".
Autre exemple :

```
1Ø  REM ... CARRE
2Ø  HIRES
3Ø  CURSET 6Ø, 4Ø, 3
4Ø  DRAW 12Ø, Ø, 1
5Ø  DRAW Ø, 12Ø, 1
6Ø  DRAW – 12Ø, Ø, 1
7Ø  DRAW Ø, –12Ø, 1
```

Remarque :
Prendre garde à ne pas donner des paramètres qui fassent sortir le curseur de l'écran, ce qui ferait apparaître un message d'erreur.

CURMOV

Syntaxe :
CURMOV X, Y, FD

But :
Cette commande déplace le curseur de sa position précédente à sa nouvelle position en X, Y sans tracer de trait (déplacement avec le "crayon levé"). Selon la valeur de FD, le point d'arrivée est marqué ou non. (Mêmes conventions que pour CURSET)

Exemple :

```
1Ø  HIRES
2Ø  CURSET 1ØØ, 1ØØ, Ø
3Ø  CIRCLE 5Ø, 1
4Ø  CURMOV 2Ø, Ø, Ø
5Ø  CIRCLE 5Ø, 1
```

Le deuxième cercle sera décalé horizontalement par rapport au premier de 2Ø points.

Remarque :
Prendre garde à ne pas donner des paramètres trop élevés ce qui ferait sortir le curseur de l'écran et ferait apparaître un message d'erreur.

DRAW

Syntaxe :
DRAW X, Y, FD

But :
Cette instruction permet de tracer un trait entre la position de point de départ du curseur et celle qui est calculée par la machine en ajoutant les valeurs de X et Y aux valeurs des coordonnées d'origine.
Autrement dit, si le curseur est en A, B, DRAW X,Y trace un trait à partir des coordonnées A, B jusqu'aux coordonnées A + X et B + Y.
FD joue le même rôle que dans CURSET.

Exemple :

```
10  REM ... CARRE
20  HIRES
30  CURSET 60, 40, 3
40  DRAW 120, 0, 1
50  DRAW 0, 120, 1
60  DRAW - 120, 0, 1
70  DRAW 0, -120, 1
```

NB : les nombres négatifs dessinent de droite à gauche et de bas en haut.

CIRCLE

Syntaxe :
CIRCLE R, FD

But :
Fonction graphique qui dessine un cercle de rayon R (1er argument) suivant le code FD qu'on lui a donné (2e argument).
FD pour FOND/DEVANT :
Ø FOND
1 DEVANT
2 INVERSION
3 RIEN
Le centre du cercle se trouve à la position initiale du curseur. Il faut donc veiller à ce que la valeur du rayon R ne fasse pas sortir le cercle de l'écran.

Exemple :

```
10  HIRES
20  CURSET 100, 100, 0
30  CIRCLE 50, 1
```

PATTERN

Syntaxe :
PATTERN X

But :
Modifie le motif tracé par DRAW ou CIRCLE.
X est le paramètre qui définit le motif. Au départ X vaut 255 ce qui correspond à un tracé continu. En changeant la valeur de X entre Ø et 255 on obtiendra des motifs différents (pointillés, tiret-point-tiret, etc.)

Exemple :

PATTERN 15 pointillé égaux
PATTERN 170 pointillés
Application au traçage d'un carré en pointillé :

```
10  REM ... CARRE EN POINTILLE ...
15  HIRES
20  PATTERN 170
40  CURSET 60, 40, 3
50  DRAW 120, 0, 1
60  DRAW 0, 120, 1
70  DRAW - 120, 0, 1
80  DRAW 0, -120, 1
```

Remarque :
Pour comprendre le calcul des paramètres, il faut savoir qu'ATMOS travaille par octets de 8 bits, c'est-à-dire par éléments de 8 cases pouvant contenir chacune un zéro ou un un. De sorte qu'il peut compter de zéro à 255 en utilisant différentes combinaisons de ces 8 cases.
255 s'écrit 11111111 (8 cases remplies par des uns) ce qui définit un motif en ligne continue pour l'instruction DRAW. Si on modifie cet octet et que l'on met à la place le nombre 15 (qui s'écrit 00001111 en binaire) seule la moitié du trait s'imprime.
Pour le calcul en binaire se reporter à la page 123

FILL

Syntaxe :
FILL A, B, C
A est le nombre de ligne à remplir. A doit être inférieur à 200 – Y si Y est le numéro de la ligne où se trouve le curseur.
B est le nombre de colonnes (ou de segments de 6 points) à remplir. B doit être inférieur à 40 – N, N étant le numéro de la colonne où se trouve le curseur.
C est le paramètre qui commande la façon dont est fait le remplissage. C doit être inférieur à 256 mais les valeurs les plus courantes sont celles inférieures à 24 où il se comporte alors comme un ATTRIBUT (Voir Annexe).

But :
En mode HIRES uniquement pour remplir une zone de l'écran pour en modifier le fond, l'encre ou pour dessiner des motifs.

Exemple :

```
10  HIRES
20  CURSET 50, 50, 0
30  FILL 40, 5, 63
```

dessine un rectangle de la couleur de l'encre à partir du point de coordonnées 50, 50. La largeur du rectangle sera de 5*6 (5 segments de 6 points) et sa hauteur de 40.
NB : FILL déplace le curseur graphique verticalement d'une valeur égale à A. Elle ne le déplace pas horizontalement.

CHAR

Syntaxe :
CHAR X,S,FD

But :
Écrit un caractère à l'emplacement du curseur. X est le code ASCII (de 32 à 127) ; S vaut 0 si on emploie le clavier standard, S vaut 1 si on emploie le second clavier ; FD vaut 0, 1, 2 ou 3 :
FD pour FOND/DEVANT
0 FOND
1 DEVANT
2 INVERSION
3 RIEN

Exemple :

```
10  HIRES
20  CURSET 120, 100, 0
30  CHAR 67, 0, 1
```

Imprime un C au milieu de l'écran graphique.

POINT

Syntaxe :
POINT (X,Y)

But :
Renvoie 0 si le point (ou pixel) indiqué est de la même couleur que le fond.

Renvoie −1 s'il est de la couleur de l'encre (de l'avant-plan).
X est un entier compris entre Ø et 239.
Y est compris entre Ø et 199

Exemple :

```
1Ø  HIRES
2Ø  CURSET 1ØØ, 1ØØ, 1
3Ø  PRINT POINT (1ØØ, 1ØØ) ; POINT (1ØØ, 1Ø1) ;
RUN
−1, Ø
Ready
```

Bruits, sons, musique
ZAP - SHOOT - PING - EXPLODE - SOUND - MUSIC - PLAY

ZAP

Syntaxe :
ZAP

But :
Cette instruction fournit un sifflement qui décroît rapidement évoquant "une arme galactique à laser".

Exemple :

```
1Ø  FOR I=1 TO 1Ø
2Ø  ZAP
3Ø  WAIT 6
4Ø  NEXT 1
```

Ce programme fournit une salve de fusil-laser. L'utilisation de WAIT, ligne 3Ø, est nécessaire pour détacher les sons.

SHOOT

Syntaxe :
SHOOT

But :
Cette instruction simule la détonation d'un fusil.

Exemple :

```
1Ø  FOR I=1 TO 1Ø
2Ø  SHOOT
3Ø  WAIT 7
4Ø  NEXT
```

Ce programme fournit une courte salve.

PING

Syntaxe :
PING

But :
Cette fonction musicale fournit un son de clochette identique à celui de la commande CTRL G.

EXPLODE

Syntaxe :
EXPLODE

But :
Cette instruction fournit un bruit d'explosion.

Exemple :

```
10  FOR I=1 TO 10
20  EXPLODE
30  WAIT 10 : REM temporisation
40  NEXT I
```

Ce programme fournit un chapelet d'explosions.
NB : l'utilisation d'une pause, ligne 30, est voulue pour détacher nettement les explosions successives. La longueur de WAIT dépend des sons.

SOUND

Syntaxe :
SOUND Canal, Période, Volume

But :
Cette instruction sonore peut être utilisée pour produire une grande variété de sons musicaux.
Tous ces paramètres sont numériques :
- *Canal :* 1, 2 ou 3 pour les sons purs
4, 5 ou 6 pour les bruits (mélange de bruits et de sons). En réalité, il n'y a qu'un canal pour les bruits : le 4, 5 ou 6 ne font que préciser quel canal son doit être mélangé au bruit.

- *Période :* ce paramètre donne la hauteur du son. Plus il est grand, plus le son est grave.
- *Volume :* 1 à 15

Si volume vaut Ø : passe à l'instruction PLAY. Le volume est alors fonction des 3e et 4e paramètres de PLAY.

Exemple :

SOUND 2, 1ØØ, 14

MUSIC

Syntaxe :
MUSIC Canal, Octave, Note, Volume

But :
Cette instruction produit des sons purs. Sa structure se prête à la composition d'airs à partir d'une partition.
- *Canal* : 1, 2 ou 3
- *Octave* : Ø à 6 (6 étant le plus grave).
- *Note* :

1	2	3	4	5	6	7	8	9	1Ø	11	12
do	do#	ré	ré#	mi	fa	fa#	sol	sol#	la	la#	si

- *Volume* : 1 à 15

Exemple :

MUSIC 3, 2, 6, 1Ø   joue un fa de la 2e octave.

PLAY

Syntaxe :
PLAY Son, Bruit, Enveloppe, Durée

But :
Cette instruction est complémentaire de MUSIC ou de SOUND
- *Son :* Ø rien ; 1 canal 1 ; 2 canal 2 ; 3 canaux 1 et 2 ; 4 canal 3 ; 5 canaux 3 et 1 ; 6 canaux 3 et 2 ; 7 canaux 3, 2 et 1.

- *Bruit :* ce paramètre est analogue à période mais agit sur les bruits.
- *Enveloppe :* manière dont le son ou le bruit est modulé : ce paramètre n'agit que si le troisième paramètre de SOUND ou MUSIC est à Ø

1 :
2 : durée fixe
3 :
4 :
5 : sons ou bruits continu
6 :
7 :

- *Durée :* Ø à 32767 ; influe sur le temps que met le son ou le bruit à commencer et à s'arrêter si l'enveloppe n'est pas périodique. Donne la période de l'enveloppe dans le cas contraire.

Exemple :

```
1Ø  MUSIC 2, 2, 4, Ø
2Ø  PLAY 2, Ø, 2, 32ØØØ
```

Le son augmente progressivement.

Gestion de la mémoire
CLEAR - DIM - RELEASE - GRAB - PEEK - POKE DOICE - DEEK - DOKE - HIMEM - CALL - DEFUSR USR

CLEAR

Syntaxe :
CLEAR

But :
Cette commande efface toutes les variablés en mémoire centrale c'est-à-dire qu'elle met les variables numériques à Ø, les variable chaînes à vide.

Exemple :

```
1Ø  B=25 : B$="VINGT-CINQ"
2Ø  PRINT B$ ;"=" ;B
3Ø  END
RUN
VINGT-CINQ=25
CLEAR
PRINT B$ ;"=" ;A
   =Ø
```

DIM

Syntaxe :
DIM nom de variable (indice)

But :
Cette instruction sert à réserver de la place pour les tableaux à une ou plusieurs dimensions. Les expressions ou indices indiquent les valeurs maxima que peuvent prendre les indices.
Lors de l'exécution, les éléments sont initialisés à la valeur Ø. Les éléments d'un tableau de chaîne le sont à la valeur "vide".
Si à l'exécution un indice est en dehors des dimensions spécifiées, l'erreur est signalée (BS).

Exemple :

DIM D(15), M(12,15)
D est un tableau à une dimension
M est un tableau à deux dimensions
DIM M$(IS), A$(12,15)
M$ et A$ sont des tableaux de chaînes.

FRE

Syntaxe :
FRE(Ø)

But :
Cette fonction donne le nombre d'octets encore disponibles dans la mémoire vive.

Exemple :

PRINT FRE(Ø)
12176 (chiffre variable, bien entendu)
FRE(X$) donne la place disponible pour le traitement des chaînes de caractères.
Remarque :
Pour nettoyer la mémoire, un truc :
A=FRE("")

RELEASE

Syntaxe :
RELEASE
But :
Cette instruction alloue la zone qui a été décrite à l'instruction GRAB à l'écran graphique.

GRAB

Syntaxe :
GRAB
But :
Permet l'utilisation de la zone mémoire graphique.

PEEK

Syntaxe :
PEEK (AD)

But :
Cette fonction fournit le contenu de l'adresse mémoire spécifiée par AD. AD est compris entre Ø et 65535. Ou Ø et FFFF en hexadécimal. Le résultat qui représente le contenu d'un octet est compris entre Ø et 255.
La fonction inverse de PEEK est POKE

Exemple :

```
1Ø  A=PEEK (1ØØØ)
1Ø  PRINT PEEK (1Ø)
48
Ready
```

POKE

Syntaxe :
POKE AD, J

But :
Cette instruction range à l'adresse AD un octet dont la valeur est donnée par J
AD et J sont des entiers.
La valeur de AD, après arrondi éventuel, doit se trouver dans l'intervalle (Ø.65535) sinon l'erreur ILLEGAL QUANTITY est signalée.
Celle de J doit se trouver entre Ø et 255 sinon sortie en erreur ILLEGAL QUANTITY.

Exemple :

```
15Ø  POKE 1ØØØ, 135
```

DEEK

Syntaxe :
DEEK (AD)
où AD représente une adresse (<65535)

But :
Cette commande fournit le contenu d'un octet augmenté de 256 fois le contenu de l'octet suivant. Lorsqu'une adresse est contenu dans deux octets consécutifs elle permet accéder directement à la valeur de l'adresse.

Exemple :

? DEEK (#DDE)

DOKE

Syntaxe :
DOKE X, V

But :
(X+1). Plus précisément elle place : INT (V/256) dans (X+1), V–INT(V/256) dans X.

Exemple :
DOKE #ABCZ, #11FF

HIMEM

Syntaxe :
HIMEM AD

But :
Dégage une zone mémoire pour des sous-programmes en langage machine. Cet emplacement est pris sur la mémoire disponible pour le BASIC.

Exemple :

100 HIMEM #8703

CALL

Syntaxe :
CALL X

But :
Branche à un sous-programme en langage machine à partir de l'adresse X.
Revient au BASIC après la rencontre du mnémonique RTS dans le programme en langage machine.

Exemple :

500 CALL #401
Appelle le sous-programme en langage machine qui débute à l'adresse hexadécimale 401.

DEFUSR USR

Syntaxe :
DEFUSR = AD
USR (Y)

But :
DEFUSR : définit l'adresse d'un sous-programme en langage machine qui sera ensuite appelé par l'expression USR.
Le paramètre Y est passé dans l'accumulateur flottant du BASIC ce qui permet de le passer au sous-programme.
Le sous-programme doit pour retourner au programme BASIC comporter comme dernière instruction #60 (Mnémonique RTS : retour de sous-programme).

Exemple :

```
10 DEFUSR #401
.............................................................
.............................................................
100 X=USR(56)
```

Le programme en langage machine devra par ailleurs être chargé par des POKE.

La sauvegarde de données sur cassette et les fichiers STORE - RECALL - FICHIERS

STORE

Syntaxe :
STORE X, ‘‘Nom du tableau’’ (,S)
X est le tableau a été déclaré au cours du programme (instruction DIM).
Par exemple, A$(4,5) ; si on a affaire à un tableau de valeurs numériques DIM : T(1ØØ).
’’nom du tableau’’ est le nom sous lequel vous décidez de l’enregistrer sur la bande magnétique ou la disquette. Ce nom ne doit pas dépasser 16 caractères. Ne pas omettre la virgule.
(,S) est l’indice facultatif de vitesse lente de sauvegarde (3ØØ bauds) ; par défaut, la vitesse sera normale (24ØØ bauds).

But :
Enregistrer un tableau de données sur bande magnétique ou sur disquette.

Exemple :

STORE A$, ‘‘BIDULE’’ (,S)
STORE T, ’’TEMPERATURE’’ (,S)
La marche à suivre est identique à celle de CSAVE. Le message SAVING, suivi du nom du tableau, apparaît en haut de l’écran. Il est accompagné d’une lettre spécifiant le type de données : R pour les nombres Réels, I pour les entiers (Integer), S pour les chaînes de caractères (Strings).
L’erreur OUT OF DATA est signalée si le tableau n’a pas été déclaré. Dans le cas où ce tableau est inférieur à 11 éléments, l’enregistrement se déroule normalement puisque le BASIC accepte implicitement de tels tableaux.
Voici deux exemples illustrant ces deux opérations de stockage et de lecture.

Enregistrement :

```
 5  REM SAUVEGARDE DE DONNÉES
10  DIM A(20)
20  FOR I=0 TO 20
30  LET A(I)=I*5
40  NEXT I
50  PRINT "APPUYEZ SUR UNE TOUCHE"
55  PRINT "DES QUE VOUS ETES PRET"
56  PRINT "A SAUVEGARDER."
60  GET A$
70  STORE A, "TABLEAU"
80  PRINT "TABLEAU ENREGISTRE".
90  END
```

Lecture :

```
 5  REM LECTURE DE DONNÉES
10  DIM B(20)
20  PRINT "APPUYEZ SUR LA TOUCHE"
22  PRINT "DU MAGNÉTOPHONE"
23  PRINT "PUIS SUR UNE TOUCHE D'ATMOS"
30  GET A$
34  PRINT : PRINT "LECTURE EN COURS"
40  RECALL B, "TABLEAU"
50  CLS
60  FOR L=0 TO 20
70  PRINT B(L)
80  NEXT L
90  END
```

RECALL

Syntaxe :

RECALL Y, "nom du fichier" (,S)

Y est le nom du tableau où vont venir prendre place les données en provenance de la cassette. Insistons sur le fait que ce tableau peut ne pas avoir le même nom que celui de la cassette mais qu'il doit être de même type et avoir au moins les mêmes dimensions.

"Nom du fichier" est le nom sous lequel les données ont été enregistrées avec STORE.

(,S) est l'indice facultatif pour la lecture en vitesse lente (Slow). A condition que le tableau ait été sauvé par STORE en vitesse lente (300 bauds).

RECALL permet de lire un tableau déjà enregistré sur cassette ou disquette. Le processus est identique à celui du chargement d'un programme BASIC par CLOAD. Auparavant, il est OBLIGATOIRE d'avoir défini et déclaré dans le programme que ces données vont alimenter, un NOUVEAU tableau du même type, de même dimension (ou plus grand), afin de recueillir les données en provenance de la cassette. Autrement dit, ce n'est pas l'image du tableau de la cassette qui est rappelée en RAM ; ce sont les données qui sont lues dans un tableau et recopiées dans un autre. L'erreur OUT OF DATA est signalée si aucun tableau n'a été déclaré pour la réception des données.

Enregistrement :

```
 5 REM SAUVEGARDE D'UN TABLEAU DE CHAÎNES
10 DIM A$(5)
20 FOR I=0 TO 5
30 INPUT B$
40 LET A$ (I)=B$
50 NEXT I
55 PRINT "APPUYEZ SUR UNE TOUCHE"
56 PRINT "DES QUE VOUS ETES PRET"
57 PRINT "A SAUVEGARDER."
60 GET A$
70 STORE A$, "TABLEAU"
80 PRINT "TABLEAU ENREGISTRE"
90 END
```

Lecture :

```
 5 REM LECTURE D'UN TABLEAU DE CHAÎNES
10 DIM B$(5)
20 PRINT "APPUYEZ SUR LA TOUCHE"
22 PRINT "MARCHE AVANT DU MAGNÉTOPHONE"
23 PRINT "PUIS SUR UNE TOUCHE D'ATMOS"
30 GET A$
34 PRINT : PRINT "LECTURE EN COURS"
40 RECALL B$, "TABLEAU"
50 CLS
60 FOR L=0 TO 5
70 PRINT B$(L)
80 NEXT L
90 END
```

ENREGISTRER DES FICHIERS

Il n'existe pas dans le BASIC de l'ORIC ATMOS d'instruction d'ouverture et de fermeture de fichiers. Mais il est possible, avec quelques notions de langage machine, d'utiliser des sous-routines déjà prêtes dans le ROM BASIC.

Gestion de fichiers sur cassette
D'abord initialiser un certain nombre de pointeur avant la sauvegarde :

Adresses	*Données*
#5F	BMS de l'adresse de début
#6Ø	BPS de l'adresse de début
#61	BMS de l'adresse de fin
#62	BPS de l'adresse de fin
#63	AUTO=1, SINON=Ø
#64	Basic=Ø, langage machine=1
#67	Vitesse : rapide=Ø, lent=1
#35	Nom du programme puis #ØØ

Par exemple, pour sauver un fichier qui se situe entre #5ØØ et #1ØØØ, le nom du fichier doit être rangé aux adresses #35, #36 et suivantes jusqu'à 17 lettres. Le dernier caractère du nom devra être #ØØ
Une fois l'initialisation terminée, vous appellerez les 3 routines suivantes :
JSR$ E6CA ; initialisation du VIA
JSR$ E57B ; sauvegarde du fichier
JSR$ E8Ø4 ; initialisation du clavier
Pour relire ce fichier, il vous suffira d'indiquer le nom du fichier, comme précédemment, puis d'exécuter les routines suivantes :
JSR$ E6CA ; initialisation du VIA
JSR$ E4A8 ; lecture du fichier
JSR$ E8Ø4 ; initialisation du clavier

Les commandes du magnéto-cassette et de l'imprimante.
CSAVE - CLOAD - LLIST - LPRINT

CSAVE

Syntaxe :
CSAVE "XX"

But :
Cette commande permet de transférer un programme (nommé ici "XX") de la mémoire vive à une cassette.
Le nom du programme doit figurer entre guillements. Le nombre de caractères autorisés peut aller jusqu'à 17, y compris les points, les traits d'union etc.

Exemple :

Brancher le magnétophone en position d'enregistrement. Taper : CSAVE "ESSAI".
Quand on appuie sur RETURN, le programme est envoyé sous forme de signaux électriques sur le ruban magnétique.
Pendant la durée du chargement, le message SAVING (en cours de sauvegarde) s'affiche en haut et à droite sur l'écran.
A la fin du chargement le mot READY (prêt) apparaît à nouveau sur l'écran.

Options :

— *Vitesse lente.*
Si on ne précise pas, la sauvegarde se fait à la vitesse de 2400 Bauds ce qui est très rapide. Si votre magnétophone n'est pas de très bonne qualité, il vaut mieux utiliser une vitesse plus lente.
Vous écrirez alors
CSAVE "XX", S
Le S indique que la sauvegarde se fera à 300 Bauds.

— *Exécution automatique.*
Si vous voulez que l'exécution se fasse automatiquement, ajoutez AUTO après le nom du programme.
CSAVE "PROG", AUTO
Après le chargement l'exécution se fait automatiquement.

— *Sauvegarde d'une zone mémoire.*
Vous pouvez sauvegarder sur cassette des blocs de mémoire (par exemple la mémoire écran). Il faut indiquer au programme les adresses de début et de fin de bloc.
La syntaxe est la suivante.
CSAVE "MEN", #AØØØ, #BFEØ sauvera l'écran graphique si vous êtes en mode HIRES.

CLOAD

Syntaxe :
CLOAD "XX"

But :
Cette commande permet de transférer un programme (nommé ici "XX") d'une cassette à la mémoire vive de l'ATMOS.
Pendant la recherche du fichier-programme sur la bande, vous lirez sur l'écran en haut : SEARCHING ... (en recherche).
Dès que le programme est trouvé ce message est remplacé par celui-ci : LOADING (chargement en cours).

Autre syntaxe :
CLOAD ""

But :
Dans le cas où l'on a oublié le nom du programme, ou que l'on veut obtenir le prochain programme. Cela peut également servir à relire systématiquement toute une cassette, à condition de retaper CLOAD"" après chaque chargement.
Un conseil : ne pas interrompre un chargement en cours car cela risquerait de nuire aux suivants.

Options :
— *Vitesse lente.*
Comme pour CSAVE, ajouter ,S après le nom du programme.
Exemple : CLOAD "SYNTHE",S
Un programme sauvé à vitesse lente doit obligatoirement être chargé à vitesse lente et inversement.
— *Exécution automatique.*
Elle se fait si elle a été demandée à la sauvegarde.
— *Chargement d'une zone mémoire.*
Même syntaxe qu'avec CSAVE :
Exemple : CLOAD "MEN", #AØØ, #BFEØ

— *Vérification d'un enregistrement.*
VERIFY est une commande qui vous permet de contrôler la sauvegarde de votre programme.
Lorsque vous avez sauvegardé votre programme, rembobinez la bande jusqu'au début de l'enregistrement.
Vérifiez que le niveau du potentiomètre du magnéto est à 80 %.
Tapez : CLOAD "BIDULE", V ou CLOAD "BIDULE", V,S (vitesse lente). Le nom du programme peut être omis si vous êtes certain de la position du programme sur la bande.
Appuyez sur RETURN.
ATMOS commence à chercher (SEARCHING). Quand il a localisé le programme, le message VERIFYING suivi du nom du programme et de son type (B pour Basic, C pour Code) apparaît dans la fenêtre supérieure.
Si l'enregistrement du programme est bon, le message "O Verify errors detected" apparaît à la position du curseur.
Si l'enregistrement s'avère défectueux, le nombre d'erreurs est indiqué dans le message. Il faut alors recommencer l'opération de sauvegarde sur une autre portion de bande vierge.

— *Enchaînement de deux programmes.*
ATMOS vous propose une autre facilité : joindre deux programmes bout à bout. Le second se place à la fin du premier déjà chargé en mémoire vive. Pour cela, positionnez votre bande au début du second programme et tapez :
CLOAD "BIDULE", J,S
Cette syntaxe empêche ATMOS de voir sa mémoire vidée de tout programme comme cela arrive lorsqu'on utilise n'importe quelle commande CLOAD. Le nouveau programme est introduit séquentiellement dans la mémoire. Pour joindre 2 programmes, il faut prendre soin de numéroter les lignes du second à partir d'un chiffre supérieur à celui de la dernière ligne du premier.
Si ce n'est pas le cas, l'exécution échouera et cela tant que les lignes des deux programmes ne s'ajusteront pas.
S'assurer également que la capacité mémoire est suffisante pour l'introduction d'un second programme.

LLIST

Syntaxe :
LLIST

But :
Sort le listing sur imprimante.
LLIST 15Ø
LLIST – 15Ø
LLIST 15Ø –
LLIST 1ØØ – 15Ø

LPRINT

Syntaxe :
LPRINT X
LPRINT X$

But :
Cette commande permet d'écrire des nombres (X) ou des chaînes de caractères (X$) sur l'imprimante.
Mêmes possibilités que PRINT.

Annexe 1
Message d'erreur

BAD SUBSCRIPT (mauvais indice)
Incompatibilité entre la déclaration d'un tableau à N dimension(s) et la variable qui l'appelle. Exemple, si l'on a déclaré DIM A$(12,6) il ne faut pas dimensionner la variable de cette façon A$(I).

BAD UNTIL (mauvais UNTIL)
Rencontre d'un UNTIL sans qu'un REPEAT ait été préalablement lu.

CAN'T CONTINUE (on ne peut pas continuer)
Votre ordinateur ne peut reprendre ou continuer l'exécution d'un programme. Deux cas : ATMOS a détecté une erreur de votre part mais vous essayez cependant de reprendre l'exécution, avec CONT ; ou bien, après avoir interrompu l'exécution d'un programme, vous l'avez modifié ou corrigé de telle sorte que cet ajout empêche l'exécution du programme.
Vous pouvez, dans ce dernier cas, reprendre l'exécution par GOTO suivi du numéro de ligne.

DISTYPE MISMATCH (incompatibilité entres modes d'écriture)
Vous êtes dans le mode TEXT et vous cherchez à ... dessiner. C'est l'exemple le plus fréquent de ce type de confusion entre les modes HIRES ou TEXT.

DIVISION BY ZERO (division par zéro)
Ce message peut provenir d'une variable qui n'a pas été initialisée et a la valeur nulle au moment d'effectuer le calcul. En effet, toute variable non initialisée explicitement a la valeur nulle dès le début.

FORMULA TOO COMPLEX (expression trop complexe)
Il peut arriver que l'expression que vous avez proposé à ATMOS soit trop compliquée. Par exemple si la ligne d'instruction contient plus de deux IF... THEN

ILLEGAL DIRECT (interdit en mode direct)
Utilisation d'une commande ou d'une instruction non utilisable en mode direct. Exemple : GET, DATA, DEF FN, INPUT...

ILLEGAL QUANTITY (valeur non admise)
Calcul de la valeur fonction en un point qui dépasse son domaine de définition. Exemple : indice de tableau supérieur à 32767 ou négatif.
Ø élevé à une puissance négative.
argument de LOG négatif ou nul.
argument de SQR négatif.

NEXT WITHOUT FOR (NEXT sans FOR)
Plusieurs raisons : boucles mal inbriquées ou bien NEXT non supprimé après l'élimination d'un FOR, lors d'une correction.

OUT OF DATA (plus de données)
Les données en DATA ont déjà été lues, on cherche pourtant à les relire. Soit repenser la logique du programme, soit utiliser l'instruction RESTORE.

OUT OF MEMORY (mémoire épuisée)
Programme trop long, trop de variable, plus de 16 boucles et GOSUB imbriqués.

OVERFLOW (dépassement de capacité)
Un nombre de valeur absolue supérieure à $1{,}7041*10^{38}$ a été trouvé dans un calcul.

REDIM'D ARRAY (tableau redimensionné)
Un tableau a déjà été dimensionné et l'on cherche à le redimensionner.

RETURN WITHOUT GOSUB (retour sans GOSUB)
Plusieurs causes : oubli de END dans le programme principal ; rencontre d'un RETURN sans qu'un GOSUB ait été préalablement lu.

STRING TOO LONG (chaînes de caractères trop longues)
Tentative de concaténer plus de 255 caractères pour former une chaîne.

SYNTAX ERROR (erreur de syntaxe)
Instruction incompréhensible pour l'interpréteur basic. Les causes sont multiples. Il vaut mieux vérifier l'instruction incriminée.

TYPE MISMATCH (incompatibilité entre variables numériques et alphanumériques).
Tentative d'attribuer une chaîne de caractères à une fonction numérique et vice versa. Tentative d'utiliser une fonction numérique avec des chaînes ou le contraire.

UNDEF'D STATEMENT (instruction non définie)
Tentative d'accéder par GOTO, GOSUB, THEN à une ligne inexistante.

UNDEF'D FUNCTION (fonction non définie)
Utilisation d'une fonction non définie par DEF FN.

REDO FROM START (recommencez au début)
Tentative, après un INPUT, de rentrer une chaîne de caractères alors qu'un nombre est attendu.

Annexe 2

Code ASCII

Code	Caractère	Codes CTRL
Ø	Nul	
1	Copie	CTRL-A
2		
3	Break	CTRL-C
4	Impression en double ligne	CTRL-D
5		
6	Déclic de touche	CTRL-F
7	Clochette (Ping)	CTRL-G
8	Espacement arrière (Curseur vers la gauche)	CTRL-H
9	Curseur vers la droite	CTRL-I
1Ø	Changement de ligne (Curseur vers le bas)	CTRL-J
11	Curseur vers le haut	CTRL-K
12	Effaçage écran	CTRL-L
13	RETURN	CTRL-M
14	Effaçage ligne	CTRL-N
15	Invalidation écran	CTRL-O
16		
17	Curseur	CTRL-Q
18		
19	Écran	CTRL-S
2Ø	Capitales (majuscules)	CTRL-T
21		
22		
23		
24	Annulation ligne	CTRL-X
25		
26		
27	ESC (changement de code)	
28		
29		
3Ø		
31		
32	Espacement	

Codes 33-127

On obtient le deuxième jeu de caractères en LORES 1. Le mode LORESØ utilise le jeu standard.

CODE	CARACTÈRE STANDARD	CARACTÈRE ALTERNÉ	CODE	CARACTÈRE STANDARD	CARACTÈRE ALTERNÉ
33	!	33	77	M	77
34	”	34	78	N	78
35	#	35	79	O	79
36	$	36	80	P	8Ø
37	%	37	81	Q	81
38	&	38	82	R	82
39	’	39	83	S	83
40	(	4Ø	84	T	84
41	)	41	85	U	85
42	*	42	86	V	86
43	+	43	87	W	87
44	,	44	88	X	88
45	–	45	89	Y	89
46	.	46	90	Z	9Ø
47	/	47	91	[	91
48	Ø	48	92	|	92
49	1	49	93	]	93
50	2	5Ø	94	↑	94
51	3	51	95	£	95
52	4	52	96	©	96
53	5	53	97	a	97
54	6	54	98	b	98
55	7	55	99	c	99
56	8	56	100	d	1ØØ
57	9	57	101	e	1Ø1
58	:	58	102	f	1Ø2
59	;	59	103	g	1Ø3
60	<	6Ø	104	h	1Ø4
61	=	61	105	i	1Ø5
62	>	62	106	j	1Ø6
63	?	63	107	k	1Ø7
64	@	64	108	l	1Ø8
65	A	65	109	m	1Ø9
66	B	66	110	n	11Ø
67	C	67	111	o	111
68	D	68	112	p	
69	E	69	113	q	
70	F	7Ø	114	r	
71	G	71	115	s	
72	H	72	116	t	
73	I	73	117	u	
74	J	74	118	v	
75	K	75	119	w	
76	L	76	120	x	

CODE	CARACTÈRE STANDARD	CARACTÈRE ALTERNÉ	CODE	CARACTÈRE STANDARD	CARACTÈRE ALTERNÉ
121	y		125	}	
122	z		126		
123	{		127	DEL	127 DEL
124	|				

Lorsque vous émettez une instruction PRINT accompagnée d'un caractère dont le code ASCII est supérieur à 128,PRINT soustrait du caractère le bit le plus fort (128 compris) et affiche directement à l'écran le code restant. Les codes ainsi "amputés" ne sont pas considérés comme des bascules de commande, mais entrés directement en tant qu'attributs. Pour afficher ces codes, on doit utiliser CHR$(i), ce qu'on ne peut faire que sur les écrans à basse résolution. Si l'on essaie de les utiliser sur l'écran HIRES, on obtiendra un message d'erreur ILLEGAL QUANTITY. Par exemple :

```
1ØØ PRINT CRH$(132);CHR$(145); "ATTRIBUT"
```

affichera la chaîne "ATTRIBUT" en lettres bleues sur fond rouge. Attention : les codes 138, 139, 142, 149, qui génèrent des caractères en hauteur double, supposent deux lignes de programme identiques pour produire l'effet désiré.

CODE	EFFET
128	premier plan noir (texte/graphiques)
129	premier plan rouge (texte/graphiques)
13Ø	premier plan vert (texte/graphiques)
131	premier plan jaune (texte/graphiques)
132	premier plan bleu (texte/graphiques)
133	premier plan magenta (texte/graphiques)
134	premier plan cyan (texte/graphiques)
135	premier plan blanc (texte/graphiques)
136	premier plan noir (texte/graphiques)
137	caractères graphiques
138	caractères en hauteur double (texte)
139	caractères en hauteur double (graphiques)
14Ø	caractères clignotants (texte)
141	caractères clignotants (graphiques)
142	caractères clignotants en hauteur double (texte)
143	caractères clignotants en hauteur double (graphique)
144	fond noir
145	fond rouge
146	fond vert
147	fond jaune
148	fond bleu
149	fond magenta
15Ø	fond cyan
151	fond blanc

Annexe 3
Code "ESCAPE" (changement de code)

Les codes "ESCAPE" suivants sont disponibles sur l'ATMOS. Ils insèrent des attributs à une position-caractère donnée de la zone mémoire d'affichage à l'écran. On peut les entrer directement à l'écran au moyen de la touche ESC suivie du caractère, ou les placer dans un programme avec l'instruction PRINT CHR$(27) suivie du caractère présenté sous forme de chaîne (soit entre guillemets, soit introduit par CHR$).

ESCAPE	0	Encre noire
ESCAPE A	1	Encre rouge
ESCAPE B	2	Encre verte
ESCAPE C	3	Encre jaune
ESCAPE D	4	Encre bleue
ESCAPE E	5	Encre magenta
ESCAPE F	6	Encre cyan
ESCAPE G	7	Encre blanche
ESCAPE H	8	Texte standard
ESCAPE I	9	Texte "graphique" ("alterné")
ESCAPE J	10	Hauteur double en standard
ESCAPE K	11	Hauteur double en "graphique"
ESCAPE L	12	Clignotement en standard
ESCAPE M	13	Clignotement en "graphique"
ESCAPE N	14	Clignotement hauteur double standard
ESCAPE O	15	Clignotement hauteur double
ESCAPE P	16	Papier noir
ESCAPE Q	17	Papier rouge
ESCAPE R	18	Papier vert
ESCAPE S	19	Papier jaune
ESCAPE T	20	Papier bleu
ESCAPE U	21	Papier magenta
ESCAPE V	22	Papier cyan
ESCAPE W	23	Papier blanc

Les codes "ESCAPE" suivants agissent sur la synchronisation de l'écran :

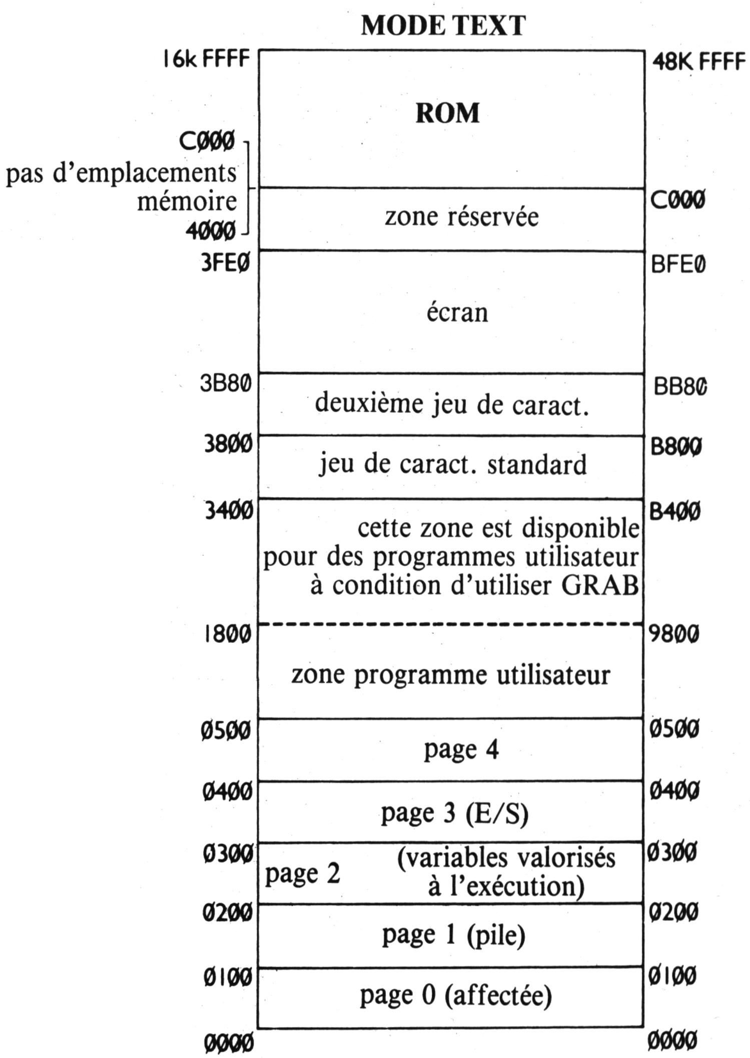
ESCAPE X	TEXT 60Hz
ESCAPE Y	TEXT 60Hz
ESCAPE Z	TEXT 50Hz
ESCAPE	TEXT 50Hz
ESCAPE	GRAPHICS 60Hz
ESCAPE	GRAPHICS 60Hz
ESCAPE	GRAPHICS 50Hz
ESCAPE —	GRAPHICS 50Hz

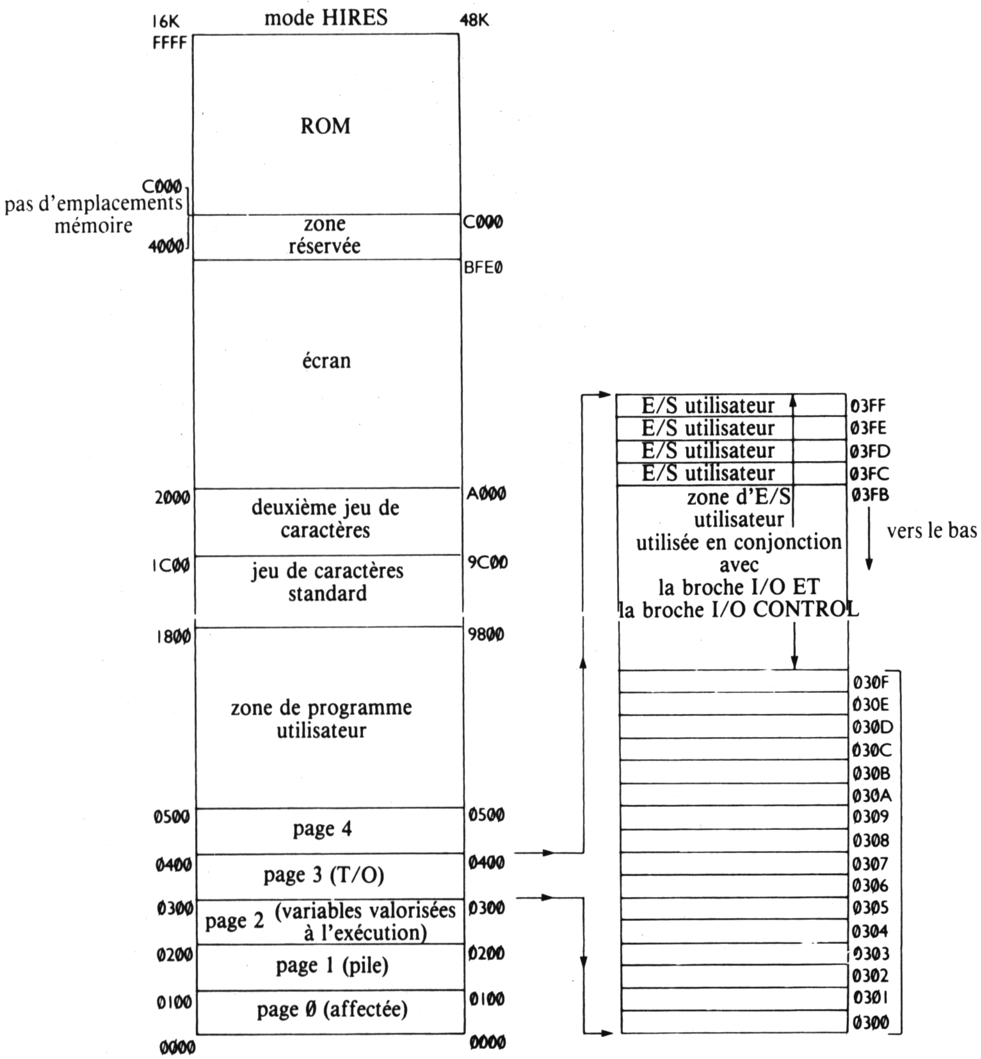
Les codes ci-dessus permettent de coordonner l'affichage à l'écran avec la fréquence du courant alternatif qui alimente l'écran vidéo ou le téléviseur servant à l'affichage de l'ATMOS : de la sorte, les signaux émis par l'ATMOS assurent les fréquences de trame correctes. La fréquence du secteur est de 5ØHz en Grande-Bretagne et, et de 6ØHz aux États-Unis et en Europe.

Annexe 4

Carte d'implantation de la mémoire

Ces plans d'implantation de la mémoire correspondent aux modes TEXT et HIRES. Toutes les adresses sont données en hexadécimal. On trouvera sur le plan HIRES les emplacements Entrée/Sortie disponibles pour l'utilisateur. Ce sont les mêmes en mode TEXT. Les adresses de l'Oric à 16 K de RAM sont données à gauche, les adresses de l'Oric 48 K à droite.





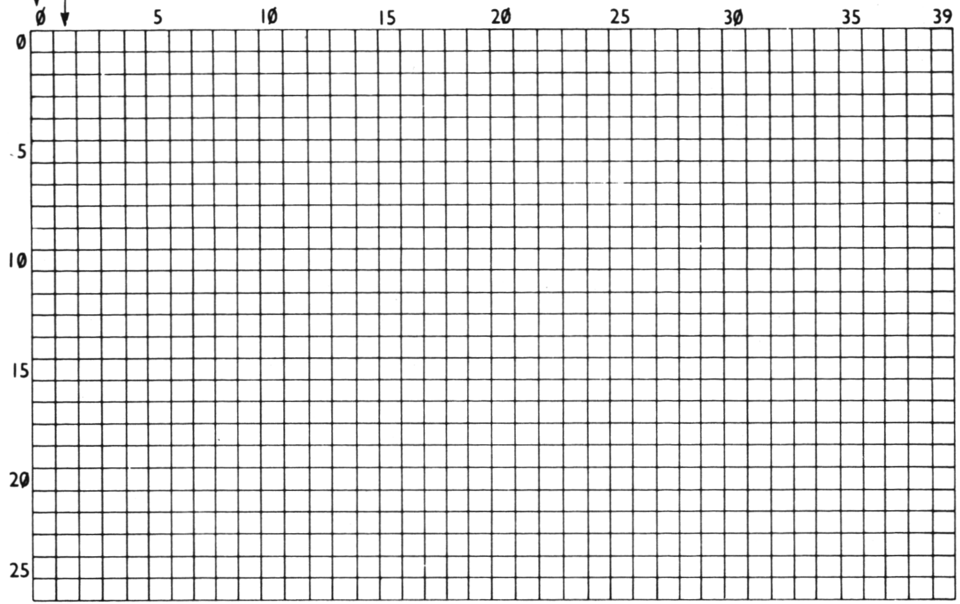
16K
mode HIRES
48K
FFFF
ROM
C000
pas d'emplacements mémoire
4000
zone réservée
C000
BFE0
écran
2000
deuxième jeu de caractères
A000
1C00
jeu de caractères standard
9C00
1800
9800
zone de programme utilisateur
0500
page 4
0500
0400
page 3 (T/O)
0400
0300
page 2 (variables valorisées à l'exécution)
0300
0200
page 1 (pile)
0200
0100
page Ø (affectée)
0100
0000
0000
E/S utilisateur
03FF
E/S utilisateur
03FE
E/S utilisateur
03FD
E/S utilisateur
03FC
zone d'E/S utilisateur utilisée en conjonction avec la broche I/O ET la broche I/O CONTROL
03FB
vers le bas
030F
030E
030D
030C
030B
030A
0309
0308
0307
0306
0305
0304
0303
0302
0301
0300

Annexe 5
Grilles d'écran

Écran TEXT

Colonne réservée (pour la couleur du fond), habituellement protégée dans les modes **TEXT** et **LORES**.

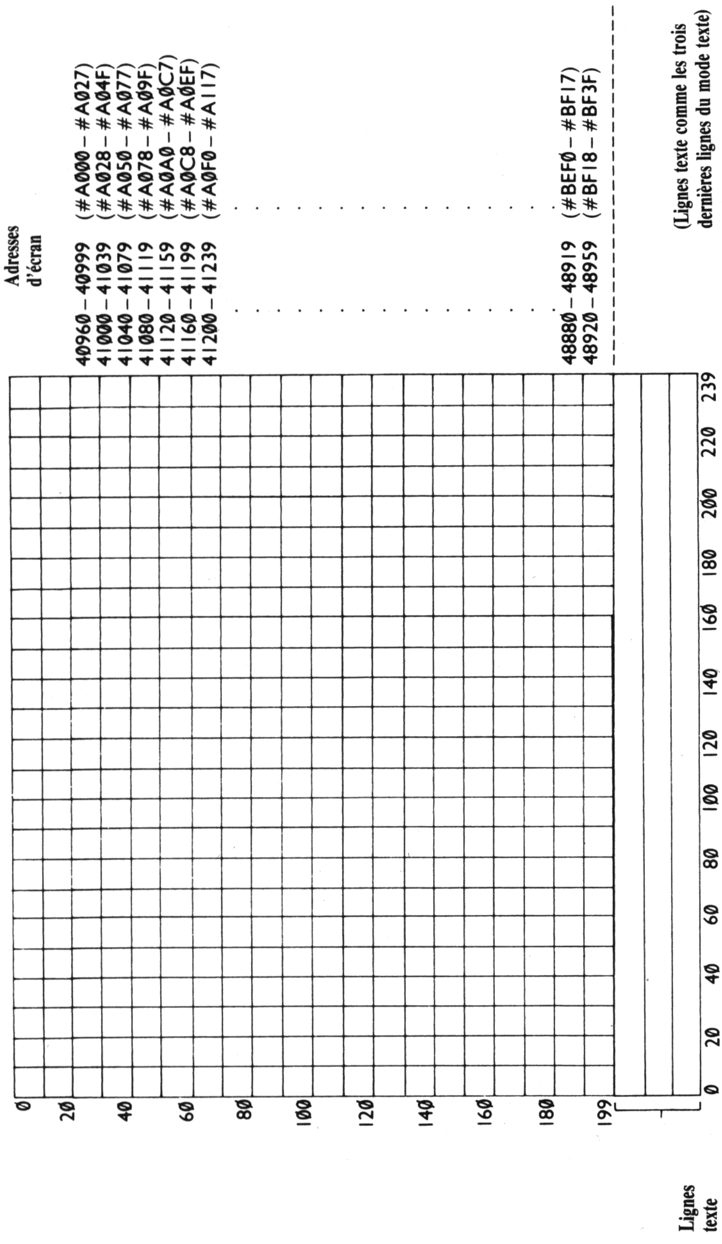
En mode **TEXT** cette colonne est normalement réservée à la couleur de premier plan : on peut l'utiliser en mode **LORES**



Coordonnées X
Mode basse résolution
(TEXT, LORES)

Adresses d'écran	
48Ø4Ø – 48Ø79	(#BBA8 – #BBCF)
48Ø8Ø – 48119	(#BBDØ – #BBF7)
4812Ø – 48159	(#BBF8 – #BC1F)
4816Ø – 48199	(#BC2Ø – #BC47)
482ØØ – 48239	(#BC48 – #BC6F)
4824Ø – 48279	(#BC7Ø – #BC97)
4828Ø – 48319	(#BC98 – #BCBF)
4832Ø – 48359	(#BCCØ – #BCE7)
4836Ø – 48399	(#BCE8 – #BDØF)
484ØØ – 48439	(#BD1Ø – #BD37)
4844Ø – 48479	(#BD38 – #BD5F)
4848Ø – 48519	(#BD6Ø – #BD87)
4852Ø – 48559	(#BD88 – #BDAF)
4856Ø – 48599	(#BDBØ – #BDD7)
486ØØ – 48639	(#BDD8 – #BDFF)
4864Ø – 48679	(#BEØØ – #BE27)
4868Ø – 48719	(#BE28 – #BE4F)
4872Ø – 48759	(#BE5Ø – #BE77)
4876Ø – 48799	(#BE78 – #BE9F)
488ØØ – 48839	(#BEAØ – #BEC7)
4884Ø – 48879	(#BEC8 – #BEEF)
4888Ø – 48919	(#BEFØ – #BF17)
4892Ø – 48959	(#BF18 – #BF3F)
4896Ø – 48999	(#BF4Ø – #BF67)
49ØØØ – 49Ø39	(#BF68 – #BF8F)
49Ø4Ø – 49Ø79	(#BF9Ø – #BFB7)
49Ø8Ø – 49119	(#BFB8 – #BFDF)

Écran HIRES


Adresses d'écran
40960 – 40999 (#A000 – #A027)
41000 – 41039 (#A028 – #A04F)
41040 – 41079 (#A050 – #A077)
41080 – 41119 (#A078 – #A09F)
41120 – 41159 (#A0A0 – #A0C7)
41160 – 41199 (#A0C8 – #A0EF)
41200 – 41239 (#A0F0 – #A117)
48880 – 48919 (#BEF0 – #BF17)
48920 – 48959 (#BF18 – #BF3F)
(Lignes texte comme les trois dernières lignes du mode texte)
0 20 40 60 80 100 120 140 160 180 199
0 20 40 60 80 100 120 140 160 180 200 220 239
Mode haute résolution (HIRES)
Lignes texte

Annexe 6

Les adresses et les routines de la ROM

Voici les routines de la ROM de l'ORIC ATMOS qui peuvent être appelées avec CALL. Elles sont également utilisables directement depuis un sous-programme écrite en langage machine par l'utilisateur.

Pour la plupart de ces routines il sera suffisant de charger le registre approprié du 65Ø2 avant d'effectuer un "JSR" à la routine désirée. Toutefois, pour les routines graphiques ou sonores, des paramètres doivent être transférés. Il faut les placer dans la zone mémoire commençant à l'adresse #2EØ. Les paramètres sont des nombres codés en complément binaire à deux sur 16 bits. Nous désignerons l'adresse des paramètres par PARAMS. En fin de sous-routine PARAMS + Ø est monté à 1 si une erreur s'est produite. Le CALL mettra à Ø PARAMS + Ø avant l'exécution de la sous-routine.

Toutes les adresses sont indiquées en hexadécimal. Toutes les routines modifient les registres A, X et Y sauf indication contraire :

VDU — adresse : F77C

Affiche le caractère à l'écran et déplace le curseur à droite.

paramètre appelé : X = caractère à afficher
paramètre renvoyé : aucun
registres affectés : aucun

STOUT — adresse : F865

Affiche un message sur la ligne d'état (adresse 48ØØØ à 48Ø39).

paramètres appelés : A = adresse du message (début)
Y = adresse du message (fin)
X = position horizontale de départ

paramètre renvoyé : X = position prochaine du curseur.

Le message est terminé par un Ø.

GTORKB — adresse : EB78

Les caractères sont renvoyés à la vitesse de répétition déterminée par le contenu des mémoires 24E et 24F.

En 24E se trouve le délai qui précède la répétition de toutes les 30 ms.

En 24F se trouve le délai entre deux affichages. Pour obtenir l'affichage le plus rapide possible monter 24E et 24F à 1.

paramètres appelés : aucun
paramètre renvoyé : A = code ASCII du caractère
regristres affectés : aucun

PRTCHR **adresse : F5C1**

Envoi d'un caractère à l'imprimante.

paramètre appelé : A = code ASCII du caractère

registres affectés : aucun

OUTLED **adresse : E75A**

En-tête (9 caractères en code ACSII n° 16, SYN) envoyé à la cassette à la vitesse prévue.

paramètres appelés : aucun
paramètres renvoyés : aucun

Noter : pour toutes les routines concernant le magnétophone la vitesse est pilotée par le contenu de l'adresse 24D. Ø pour rapide (2400 bauds). Plus grand que Ø pour lent (300 bauds).

GETSYN **adresse : E735**

Lit les octets parvenant du magnétophone jusqu'au moment de la synchronisation.

paramètres appelés : aucun
paramètres renvoyés : aucun
registres affectés : A et X

OUTBYT **adresse : E65E**

Émet un octet vers le magnétophone à la vitesse choisie.

paramètre appelé : A = caractère à émettre
paramètres renvoyés : aucun
registre affecté : A

RDBYTE **adresse : E6C9**

Lit un octet parvenant du magnétophone à la vitesse choisie.

paramètres appelés : aucun
paramètre renvoyé : A = caractère reçu
registre affecté : A

CURSET **adresse : FØC8**

paramètres appelés : PARAMS + 1 : valeur de X
PARAMS + 3 : valeur de Y
PARAMS + 5 : valeur de FD

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

CURMOV **adresse : FØFD**

paramètres appelés : PARAMS + 1 : valeur de X
PARAMS + 3 : valeur de Y
PARAMS + 5 : valeur de FD

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

DRAW **adresse : F11Ø**

paramètres appelés : PARAMS + 1 : valeur de X
PARAMS + 3 : valeur de Y
PARAMS + 5 : valeur de FD

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

CHAR **adresse : F12D**

paramètres appelés : PARAMS + 1 : code ASCII du caractère
PARAMS + 3 : clavier
Ø standard
1 semi-graphique
PARAMS + 5 : valeur de FD
paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

CIRCLE **adresse : F37F**

paramètres appelés : PARAMS + 1 : rayon
PARAMS + 3 : valeur de FD

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

PATRN **adresse : F11D**

Pattern
paramètres appelés : PARAMS + 1 : de Ø à 255
paramètres renvoyés : PARAMS mis à 1 si la valeur passé est négative ou supérieure à 255.
registre affecté : X

POINT **adresse : F1C8**

paramètres appelés : PARAMS + 1 : valeur de X
PARAMS + 3 : valeur de Y
paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas à un point de l'écran graphique.
PARAMS + 1 mis à Ø = couleur du fond.
PARAMS + 1 mis à 1 = couleur de l'avant-plan.

FILL **adresse : F268**

paramètres appelés : PARAMS + 1 : nombre de lignes
PARAMS + 3 : nombre de cellules (de 6 pixels)
PARAMS + 5 : de Ø à 255

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

PAPER **adresse : E2Ø4**

paramètre appelé : PARAMS + 1 : de Ø à 7 (couleur)
paramètres renvoyés : PARAMS mis à 1 si la valeur passée ne convient pas.

INK **adresse : F21Ø**

paramètre appelé : PARAMS + 1 : Ø à 7 (couleur)
paramètres renvoyés : PARAMS mis à 1 si la valeur passée ne convient pas.

PING accès direct **adresses FA9F**

SHOOT **FAB5**
EXPLD **FACB**
ZAP **FAE1**

KB BEEP accès direct **adresse : FB14**

Produit le "BIP" du clavier
(petits caractères)

CONTBP accès direct **adresse : FB2A**

Produit le "BIP" de la touche CTRL
RETURN, ESC.

SOUND **adresse : FB4Ø**

paramètres appelés : PARAMS + 1 : canal (de 1 à 6)
PARAMS + 3 : période (de 1 à 65535)
PARAMS + 5 : volume (de Ø à 15)

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

MUSIC **adresse : FC18**

paramètres appelés : PARAMS + 1 : canal (de 1 à 3)
PARAMS + 3 : octave (de Ø à 7)
PARAMS + 5 : note (de Ø à 12)
PARAMS + 7 : volume (de Ø à 15)

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

PLAY **adresse : FBDØ**

paramètres appelés : PARAMS + 1 : canaux actifs (Ø à 7) (son)
PARAMS + 3 : canaux actifs (Ø à 7) (bruit)
PARAMS + 5 : enveloppe (1 à 7)
PARAMS + 7 : durée enveloppe (Ø à 65535)

paramètres renvoyés : PARAMS mis à 1 si les valeurs passées ne conviennent pas.

W8912 **adresse : F59Ø**

Va lire en A l'adresse du 8912 où sera recopié le contenu de X. La routine vérifie en outre que la lecture du clavier n'est pas inhibée. Le registre OE ne doit pas être utilisé car c'est le port externe utilisé par le clavier.

paramètres appelés : A = numéro du registre de 8912
X = donnée à transférer
paramètres renvoyés : aucun

Annexe 7

RAM DE L'ORIC ATMOS (page zéro/page deux)

PAGE Ø

LINWID	# 31	longueur de la ligne pour le terminal (longueur de la ligne de l'imprimante en version 1.Ø) (4Ø par défaut).
TXTTAB	# 9A- # 9B	début du texte Basic
VARTAB	# 9C- # 9D	début des variables
ARYTAB	# 9E- # 9F	début des tableaux
STREND	# AØ- # A1	fin des variables, LOMEM
MEMSIZ	# A6- # A7	sommet de la mémoire disponible, HIMEM
CHRGET	# E2- # E7	code pour incrémenter TXTPTR
CHRGOT	# E8	instruction LDA
TXTPTR	# E9- # EA	pointeur indiquant le prochain caractère à interpréter.
SKPSPC	# EB- # EE	envoie à CHRGET en cas d'appui sur la barre d'espace.
QNUM	# EF- # F8	retenue si Ø - 9 et indicateur de zéro si CHR$(Ø).
CHRRTS	# F9	instruction de retour.

PAGE 2

KEYAD	# 2Ø8	adresse de la dernière touche enfoncée.
KBSTAT	# 2Ø9	# A4 = SHIFT de gauche # A7 = SHIFT de droite } prioritaires # A2 = CTRL # A5 = FUNCT
CAPLCK	# 2ØC	# FF = CAPS # 7F = minuscules.
PAT	# 213	registre pour Pattern (CIRCLE/DRAW)
CURX	# 219	position horizontale du curseur en HIRES
CURY	# 21A	position verticale du curseur en HIRES
GRA	# 21F	1 = HIRES, Ø = TEXT ou LORES.
SXTNK	# 22Ø	1 = 16K sinon 48K
XVDU	# 238	saut à la routine VDU
XGETKY	# 23B	saut à la routine GTORKB
XPRTCH	# 23E	saut à la routine PRTCHR
XSTOUT	# 241	saut à la routine STOUT
INTFS	# 244	saut pour interruption
NMIJP	# 247	saut à la routine d'interruption non masquable (NMI)

INTSL	#24A	retour après interruption. (normalement RTI mais possibilité de l'associer à un saut).
TSPEED	#24D	vitesse : Ø rapide, différent de Ø : lent,
	#24E	temporisation pour répétition automatique des touches du clavier.
KBRPT	#24F	cadence de répétition des touches.
PWIDTH	#256	nombre de colonnes sur l'imprimante (8Ø par défaut).
VWIDTH	#257	nombre de colonnes à l'écran (4Ø par défaut).
CURROW	#268	position verticale du curseur (n° de la ligne).
MODE Ø	#26A	les bits de cette octet définissent l'état de diverses fonctions.

Annexe 8 — Code opératoire du 6502

LANGAGE MACHINE

Mnémonique	Fonction	Langage d'assemblage	Mode d'adressage	Nombre d'octets	Code HEX	Drapeaux
ADC Addition du contenu mémoire à l'accumulateur avec retenue	A ← A+M+C	ADC #opé ADC opér ADC opér, X ADC opér ADC opér, X ADC opér, Y ADC (opér, X) ADC (opér), Y	immédiat zéro-page zéro-page, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	2 2 2 3 3 3 2 2	69 65 75 6D 7D 79 61 71	NVZC
AND ET logique entre accumulateur et mémoire	A ← A∧M	AND #opér AND opér AND opér, X AND opér AND opér, X AND opér, Y AND (opér, X) AND (opér), Y	immédiat zéro-page zéro-page, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	2 2 2 3 3 3 2 2	29 25 35 2D 3D 39 31 31	NZ
ASL Décalage d'un cran à gauche (0 dernier bit à droite) (le bit de gauche va dans C)		ASL A ASL opér ASL opér, X ASL opér ASL opér, X	accumulateur zéro-page zéro-page, X absolu absolu, X	1 2 2 3 3	0A 06 16 0E 1E	NZC
BCC Branchement si C = 0	Br. si C = 0	BCC opér	relatif	2	90	
BCS Branchement si C = 1	Br. si C = 1	BCS opér	relatif	2	B0	
BEQ Branchement si Z = 1	Br. si Z = 1	BEQ opér	relatif	2	F0	
BIT Comparaison bit par bit	$Z \leftarrow \overline{A \wedge M}$ $N \leftarrow M_7$ $V \leftarrow M_6$	BIT * opér BIT * opér	zéro-page absolu	2 3	24 2C	NVZ
BMI Branchement si N = 1	Br. si N = 1	BMI opér	relatif	2	30	
BNE Branchement si Z = 0 (résultat non nul)	Br. si Z = 0	BNE opér	relatif	2	D0	
BPL Branchement si positif ou nul	Br. si N = 0	BPL opér	relatif	2	10	
BRK Arrêt forcé	empile PC + 2 et P	BRK*	implicite	1	00	B I ← 1
BVC Branchement si non débordement	Br. si V = 0	BVC opér	relatif	2	50	
BVS Branchement si débordement	Br. si V = 1	BVS opér	relatif	2	70	
CLC Annulation de retenue	C ← 0	CLC	implicite	1	18	C ← 0
CLD Annulation du mode décimal	D ← 0	CLD	implicite	1	D8	D ← 0
CLI Autorisation des interruptions.	I ← 0	CLI	implicite	1	58	I ← 0
CLV Annuler le drapeau de débordement	V ← 0	CLV	implicite	1	B8	V ← 0

Mnémonique	Fonction	Langage d'assemblage	Mode d'adressage	Nombre d'octets	Code HEX	Drapeaux
CMP						N Z C
Comparer mémoire	A – M	CMP # opér	immédiat	2	C9	
et accumulateur		CMP opér	zéro-page	2	C5	
		CMP opér, X	zéro-page, X	2	D5	
		CMP opér	absolu	3	CD	
		CMP opér, X	absolu, X	3	DD	
		CMP opér, Y	absolu, Y	3	D9	
		CMP (opér, X)	(indirect, X)	2	C1	
		CMP (opér), Y	(indirect), Y	2	D1	
CPX	X – M					N Z C
Comparer mémoire et		CPX # opér	immédiat	2	E0	
registre X		CPX opér	zéro-page	2	E4	
		CPX opér	absolu	3	EC	
CPY						N Z C
Comparer mémoire et	Y – M	CPY # opér	immédiat	2	C0	
registre Y		CPY opér	zéro-page	2	C4	
		CPY opér	absolu	3	CC	
DEC						N Z
Diminuer de 1 le contenu	M ← M – 1	DEC opér	zéro-page	2	C6	
de la mémoire		DEC opér, X	zéro-page, X	2	D6	
		DEC opér,	absolu	3	CE	
		DEC opér, X	absolu, X	3	DE	
DEX						N Z
Diminuer de 1 le contenu	X ← X – 1	DEX	implicite	1	CA	
de X.						
DEY						N Z
Diminuer de 1 le contenu	Y ← Y – 1	DEY	implicite	1	88	
de Y						
EOR						N Z
Ou exclusif entre	A ← A V M	EOR # opér	immédiat	2	49	
mémoire et accumulateur		EOR opér	zéro-page	2	45	
		EOR opér, X	zéro-page, X	2	55	
		EOR opér,	absolu	3	4D	
		EOR opér, X	absolu, X	3	5D	
		EOR opér, Y	absolu, Y	3	59	
		EOR (opér, X)	(indirect, X)	2	41	
		EOR (opér), Y	(indirect), Y	2	51	
INC						N Z
Incrémenter de 1 la	M ← M + 1	INC opér	zéro-page	2	E6	
mémoire		INC opér, X	zéro-page, X	2	F6	
		INC opér	absolu	3	EE	
		INC opér, X	absolu, X	3	FE	
INX						N Z
Incrémenter X de 1	X ← X + 1	INX	implicite	1	E8	
INY						N Z
Incrémenter Y de 1	Y ← Y + 1	INY	implicite	1	C8	
JMP						
Saut inconditionnel	PCL ← (PC+1)	JMP opér	absolu	3	4C	
à une adresse	PCH ← (PC+2)	JMP (opér)	indirect	3	6C	
JSR	Empile PC+2	JSR opér	absolu	3	20	
Saut inconditionnel	PCL ← (PC+1)					
à un sous programme	PCH ← (PC+2)					
LDA						N Z
Charge l'accumulateur	A ← M	LDA # opér	immédiat	2	A9	
avec la mémoire		LDA opér	zéro-page	2	A5	
		LDA opér, X	zéro-page, X	2	B5	
		LDA opér,	absolu	3	AD	
		LDA opér, X	absolu, X	3	BD	
		LDA opér, Y	absolu, Y	3	B9	
		LDA (opér, X)	(indirect, X)	2	A1	
		LDA (opér), Y	(indirect), Y	2	B1	

Mnémonique	Fonction	Langage d'assemblage	Mode d'adressage	Nombre d'octets	Code HEX	Drapeaux
LDX Charger x avec la mémoire	X ← M	LDX #opér LDX opér LDX opér, Y LDX opér LDX opér, Y	immédiat zéro-page zéro-page, Y absolu absolu, Y	2 2 2 3 3	A2 A6 B6 AE BE	N Z
LDY Charger Y avec la mémoire	Y ← M	LDY #opér LDY opér LDY opér, X LDY opér LDY opér, X	immédiat zéro-page zéro-page, X absolu absolu, X	2 2 2 3 3	A0 A4 B4 AC BC	Z
LSR Décalage d'un cran à droite (0 entre à gauche) (le bit de droite va dans C)		LSR A LSR opér LSR opér, X LSR opér LSR opér, X	accumulateur zéro-page zéro-page, X absolu absolu, X	1 2 2 3 3	4A 46 56 4E 5E	Z C N ← 0
NOP Instruction muette	aucune	NOP	implicite	1	EA	
ORA OU inclusif entre mémoire et accumulateur	A ← AVM	ORA #opér ORA opér, ORA opér, X ORA opér ORA opér, X ORA opér, Y ORA (opér, X) ORA (opér), Y	immédiat zéro-page zéro-page, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	2 2 2 3 3 3 2 2	09 05 15 0D 1D 19 01 11	N Z
PHA Empiler A	empile A S ← S − 1	PHA	implicite	1	48	
PHP Empiler P	Empile P S ← S − 1	PHP	implicite	1	08	
PLA Dépiler A	A ← (Pile) S ← S + 1	PLA	implicite	1	68	N Z
PLP Dépiler P	P ← (Pile) S ← S − 1	PLP	implicite	1	28	rétablis
ROL Permutation circulaire d'un cran à gauche (le bit de C entre à droite) (le bit de gauche va dans C)		ROL A ROL opér ROL opér, X ROL opér, ROL opér, X	accumulateur zéro-page zéro-page, X absolu absolu, X	1 2 2 3 3	2A 26 36 2E 3E	N Z C
ROR Permutation circulaire d'un cran à droite (le bit de C entre à gauche) (le bit de droite va dans C)		ROR A ROR opér ROR opér, X ROR opér ROR opér, X	accumulateur zéro-page zéro-page, X absolu absolu, X	1 2 2 3 3	6A 66 76 6E 7E	N Z C
RTI Retour après interruption	P ← (Pile) S ← S + 1 PCL ← (Pile) S ← S + 1 PCH ← (Pile) S ← S + 1	RTI	implicite	1	40	rétablis

Mnémonique	Fonction	Langage d'assemblage	Mode d'adressage	Nombre d'octets	Code HEX	Drapeaux
RTS Retour après un sous-programme	PC ← (Pile) PC ← PC + 1 S ← S + 2	RTS	implicite	1	60	
SBC Soustraction du contenu de la mémoire à l'accumulateur avec retenue	A←A–M–C	SBC# opér SBC opér SBC opér, X SBC opér SBC opér, X SBC opér, Y SBC (opér, X) SBC (opér), Y	immédiat zéro-page zéro-page, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	2 2 2 3 3 3 2 2	E9 E5 F5 ED FD F9 E1 F1	N V Z C
SEC Mettre à 1 la retenue	C ← 1	SEC	implicite	1	38	C ← 1
SED Mettre en mode décimal	D ← 1	SED	implicite	1	F8	D ← 1
SEI Interdiction des interruptions	I ← 1	SEI	implicite	1	78	I ← 1
STA Recopier dans la mémoire le contenu de l'accumulateur	M ← A	STA opér STA opér, X STA opér STA opér, X STA opér, Y STA (opér, X) STA (opér), Y	zéro-page zéro-page, X absolu absolu, X absolu, Y (indirect, X) (indirect), Y	2 2 3 3 3 2 2	85 95 8D 9D 99 81 91	
STX Recopier en mémoire le contenu de X	M ← X	STX opér STX opér STX opér, Y	absolu zéro-page zéro-page, Y	3 2 2	8E 86 96	
STY Recopier en mémoire le contenu de Y	M ← Y	STY opér STY opér STY opér, X	absolu zéro-page zéro-page, X	3 2 2	8C 84 94	
TAX Transférer A dans X	X ← A	TAX	implicite	1	AA	N Z
TAY Transférer A dans Y	Y ← A	TAY	implicite	1	A8	N Z
TSX Transférer S dans X	X ← S	TSX	implicite	1	BA	N Z
TXA Transférer X dans l'accumulateur.	A ← X	TXA	implicite	1	8A	N Z
TXS Transférer X dans S	S ← X	TXS	implicite	1	9A	
TYA Transférer Y dans A	A ← Y	TYA	implicite	1	98	N Z

LISTE DES PROGRAMMES

*Ces programmes sont ceux qui figurent sur la cassette d'accompagnement.

STARTER 1
STARTER 2
STARTER 3
STARTER 4
SUPER START*
RAYON
COUP DE DE 1
COUP DE DE 2*
ANAGRAMME*
BONJOUR
SENTIMENT
DIALOGUE*
LEXIQUE*
VERTICAL
ECHEANCE
CHANGEMENT D'ADRESSE
CODE ASCII
DECODAGE
DECOMPTE DES VOYELLES*
ALPHABET*
CARRE
RECTANGLES*
EDITEUR DE DESSIN*
ORGUE*
SONS*
MUSIC
TEMPERATURE
TEMPERATURE ET MOYENNE*
TEMPERATURE 3
BATAILLE NAVALE*
CLIGNOTEMENT DOUBLE
CLIGNOTEMENT LOCALISE
TRI DE NOMBRE PAR ORDRE DECROISSANT*
CARNET D'ADRESSE*
CADAVRE EXQUIS*
SEMAINE
REPETITEUR 1*
REPETITEUR 2*
CARNET 2
LETTRES*
RIORIM EMARGORP*
RIEN*
SINUSOÏDE
COURBE*
CUBE*
SYNTHETISEUR*
OREILLE*
RIORIM 2
CHAMP DE FORCE

INDEX GÉNÉRAL

Les chiffres entre parenthèses renvoient aux fiches de référence.

ACHEVÉ D'IMPRIMER
LE 29 AOUT 1984
SUR LES PRESSES
DE
L'IMPRIMERIE
CARLO DESCAMPS
A CONDÉ-SUR-L'ESCAUT
59163 FRANCE

Dépôt légal : septembre 1984
N° d'imprimeur : 3541
Imprimé en France

Voici un guide complet pour tous. De "l'embarquement" à la "croisière hyper-basic", le pilotage de l'ATMOS (1re partie) vous propose une initiation progressive au langage basic. Vous effectuez ensuite vos premiers pas en langage machine. Cinquante programmes de jeux, vie pratique, gags, musique et dessins soutiennent votre activité.

"ATMOS en fiches" (2e partie) vous présente ensuite des références faciles à consulter et une importante documentation en annexe. Vous serez ainsi en mesure d'utiliser au maximum la puissance de la mémoire, la richesse et la souplesse de programmation de ce remarquable micro-ordinateur.

ISBN 2-7124-**0585**-4

78,50/84/09

GUIDE PRATIQUE DE L'ORIC-ATMOS

du Basic à l'Assembleur

cedic/nathan